Warsurfer

C.Wider und O.Schacher

14. Dezember 2005

Inhaltsverzeichnis

I Einführung **4**

1 Kurzfassung der Diplomarbeit **5**

2 Management Summary **6**

3 Aufgabenstellung **8**
3.1 Einführung 8
3.2 Aufgabe 9
3.2.1 Randbedingungen und Infrastruktur 9

4 Grundlagen **10**
4.1 Wireless Lan 802.11 10
4.2 Wardriving 12

II Systementwicklung **13**

5 Leistungsmerkmale **14**

6 Analyse **17**
6.1 Information Gathering 17
6.2 Kenngrössen der Netzwerkerkennung 19
6.3 Assoziation 20
6.4 Default settings 23
6.5 Netzverfügbarkeit 24
6.6 Rechtliche Einschränkungen 24
6.7 Verbindungsunterbrüche 26
6.8 Kenngrössen eines Verbindungsalgorithmus 27
6.9 Roaming 27
6.9.1 Autonomer Client 28
6.9.2 Home Proxy 28
6.9.3 MobileIP 29
6.10 Hardware 30

7 Design **31**
7.1 Testumgebung 31
7.1.1 OpenWRT 31
7.1.2 HostAP 31
7.1.3 Scripting Engine 31

7.1.4 Simulation 33
7.1.5 Feldtest 33
7.2 Applikationsdesign 33
7.2.1 Interprozesskommunikation 33
7.2.2 Klassen Design 34
7.2.3 Datentransfer 43
7.3 Datenbank 45
7.3.1 Tabellen 46
7.3.2 Sichten 49
7.3.3 Funktionen 51

III Systemanwendung 53

8 Warsurfer HOW-TO 54
8.1 Über die Warsurfer Installation 54
8.2 Abhängigkeiten 54
8.2.1 Python 55
8.2.2 dnet 56
8.2.3 scapy 56
8.2.4 Postgresql 57
8.2.5 psycopg 57
8.2.6 Kismet 57
8.3 Installation 58
8.3.1 Datenbank(automatisch) 58
8.3.2 Datenbank(Manuell) 60
8.3.3 Startskript 62
8.4 Konfiguration 62
8.4.1 Kismet 63
8.4.2 Warurfer 63
8.5 Anwendung 68
8.5.1 Zusammengefasster Start 68
8.5.2 Einzelstart 68
8.6 Download grösserer Dateien 71
8.7 Mailempfang 72
8.8 Surfen mit Openvpn 73
8.8.1 Installation 74
8.8.2 Konfiguration 74
8.9 Tweaks 76
8.9.1 Firefox 76
8.9.2 SSH 76
8.9.3 DNS 77

9 Tests 78
9.1 Netzerkennung 78
9.2 Verbindung 80
9.3 Maildownload 81
9.4 Feldtest 82

IV Bewertung und Ausblick **84**

10 Resultate und Zielerreichung **85**
10.1 Zielerreichung gemäss Aufgabenstellung 85
10.2 Feldtest 86
10.3 Soll-Ist Vergleich 87

11 Persönliche Eindrücke **88**
11.1 C. Wider 88
11.2 O. Schacher 89

12 Ausblick **90**
12.1 Verbleibende Probleme 90
12.2 Weiterentwicklung 91

13 Dank **93**

V Appendix **94**

14 Risikoanalyse **95**

15 Projektmanagementplan **97**
15.1 Einführung 97
15.1.1 Projektübersicht 97
15.1.2 Projekt Lieferumfang 97
15.1.3 Referenzliste 98
15.2 Projekt Organisation 98
15.2.1 Prozess Modell 98
15.2.2 Organisation: Abgrenzung und Schnittstellen 99
15.2.3 Projektverantwortlichkeiten 99
15.3 Projektmanagement 99
15.3.1 Management Ziele und Prioritäten 99
15.3.2 Voraussetzungen, Abhängigkeiten und Randbedingungen 99
15.3.3 Risiko Management 99
15.3.4 Überwachung und Kontrolle 99
15.4 Entwicklungsprozess 100
15.4.1 Methoden, Tools und Techniken 100
15.4.2 Software Dokumentation 100
15.5 Entwicklungsplan 100
15.5.1 1. Iteration 100
15.5.2 2. Iteration 102
15.5.3 Abhängigkeiten 102
15.5.4 Termine 102

16 Projektplan **103**

Glossar **107**

Teil I

Einführung

Kapitel 1

Kurzfassung der Diplomarbeit

Abteilung	Informatik
Name der Diplomanden	C. Wider und O. Schacher
Diplomjahr	2006
Titel der Diplomarbeit	Warsurfer
Examinatorin / Examinator	Prof. Dr. A. Steffen

Aufgabe: In dicht besiedelten Gebieten gibt es nach wie vor sehr viele unverschlüsselte Drahtlosnetzwerke, welche den Zugang zum Internet erlauben. Dieses Projekt sollte es ermöglichen, diese Netzwerke zu erkennen und mit einem ausgeklügelten Algorithmus möglichst schnell darauf zu verbinden. Mit dieser Grundlage sollte es möglich sein, während einer Busfahrt so oft und so lange wie möglich eine Internetverbindung herzustellen. In einem weiteren Schritt sollte eine Methode implementiert werden, mit den vielen Netzunterbrüchen und IP-Wechseln umzugehen, so dass zumindest das Lesen der Email-Nachrichten ermöglicht wird. Da man Software nur schlecht im Bus entwickeln kann, sollte zudem eine Testumgebung aufgebaut werden, welche diese oft wechselnden Netze simuliert.

Lösung: Der Clou des Verbindungsalgorithmus besteht darin, die IP-Settings eines Accesspoints zu erraten, um den langsamen DHCP-Request zu vermeiden. Dies geschieht mittels verschiedener Anhaltspunkte wie MAC-Adresse, ESSID oder gesniffter Pakete. Der Transfer der Emails wurde über HTTP realisiert, da dies die Fortsetzung unterbrochener Downloads unterstützt. Zudem wurde eine Lösung mit einer VPN Software realisiert, welche ein vollständiges Roaming auf IP-Ebene erlaubt, d.h. es ist möglich, bestehende TCP-Verbindungen aufrecht zu erhalten. Mit OpenWRT Accesspoints und HostAp Netzwerktreiben konnte eine Testumgebung aufgebaut werden, welche die automatisierte Simulation ständig wechselnder Wireless-Netzwerke erlaubt.
Der Softwaredesign ist stark Datenbankbasiert. Alle gefundenen Netze werden kategorisiert, um die besten Kandidaten zu erkennen und detaillierte Auswertungen und Statistiken zu erstellen.

Kapitel 2

Management Summary

Die Verbreitung von Drahtlosnetzwerken nimmt immer weiter zu und so gibt es an dicht besiedelten Orten praktisch keinen Punkt mehr, an dem nicht mindestens ein Netzwerk verfügbar ist. Nach wie vor ist ein hoher Prozentsatz dieser Netzwerke unverschlüsselt und bietet einen freien Zugriff ins Internet. Die Idee war es nun, eine Software zu entwickeln, welche diese Netzwerke entdeckt und so schnell wie möglich eine Verbindung ins Internet herstellt, um es dem Anwender zu ermöglichen, sogar während einer Busfahrt seine Email-Nachrichten zu lesen, ohne das dadurch Kosten anfallen. Diverse Schwierigkeiten (lange Assoziationszeit, problematische Hardware etc.) mussten für die Lösung dieses Problem analysiert werden. Um die Software zu entwickeln, musste zudem eine Testumgebung aufgebaut werden, welche die Gegebenheiten simulieren kann.

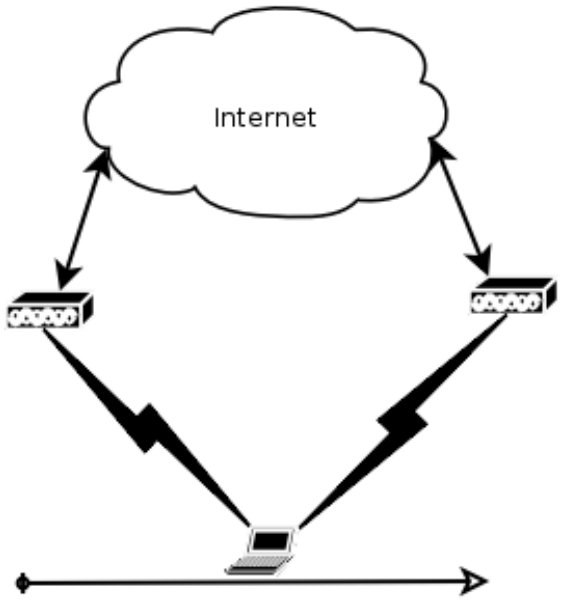


Abbildung 2.1: Warsurfer Theorie [Schematisch]

Die Anforderungen des Projekts konnten erfüllt und teilweise sogar übertroffen werden. Es war uns möglich, während einer Testfahrt in Zürich Webseiten zu lesen und sogar Email-Nachrichten aus dem fahrenden Tram zu verschicken. Der Erfolgsfaktor einer Fahrt hängt aber sehr stark von der gefahrenen Strecke und der verwendeten Hardware ab. Die Software ist relativ komplex in der Installation und Anwendung und wird daher eher von technisch versierten Personen verwendet werden.
Die der Software zugrunde liegende Datenbank ermöglicht detaillierte Statistiken und gibt einen Aufschluss über das aktuelle IT-Sicherheitsbewusstein der Bevölkerung.

Kapitel 3

Aufgabenstellung

Personen

Studenten Cédric Wider, Oli Schacher

Betreuer Prof. Dr. Andreas Steffen

Termine

Beginn Montag, 24. Oktober 2005

Abgabe Freitag, 16. Dezember 2005

3.1 Einführung

Wardriving ist ein unter Geeks beliebtes Hobby. Dabei wird versucht, von einem Laptop aus über fremde, meist ungeschützte Funknetzwerke eine Verbindung ins Internet zu erhalten. Nach wie vor sind viele Wireless-Netzwerke nicht gesichert, sei dies, weil sich Benutzer des Sicherheitsrisikos nicht bewusst sind, das technische Know-How fehlt oder schlicht weil der Betreiber des Accesspoints einen öffentlichen Internet-Gateway zur Verfügung stellen möchte. Ein solcher Accesspoint kann selbst von unerfahrenen Leuten benutzt werden. Der Nachteil dabei ist aber, dass man geographisch an denjenigen Accesspoint gebunden ist, mit dem man sich gerade verbunden hat. Bewegt man sich zu weit von diesem Verbindungspunkt weg, verschlechtert sich das Signal zunehmends und sämtliche Verbindungen ins Internet brechen schlussendlich ab.
Vor allem in dicht besiedelten Gebieten ist die Konzentration von offenen Wireless-Netzwerken mittlerweile so hoch, dass man versuchen könnte, Daten auch mobil, d.h. z.B. während einer Busfahrt zu transferieren. Das Problem hierbei ist, dass die Zeit, in der man sich in der Nähe eines Accesspoints befindet meist nicht reicht, um eine manuelle Verbindung herzustellen, geschweige denn ganze Email-Nachrichten herunterzuladen.
Die Idee hinter dieser Diplomarbeit ist es nun, ein Programm zu entwickeln, welches den Prozess der Netzsuche und der Verbindung automatisiert. Mit Hilfe

eines sinnvollen Algorithmus soll so schnell wie möglich das “beste” Netz ermittelt werden, um sich mit diesem zu verbinden. Dadurch ist je nach Umgebung eine mehr oder weniger stetige Verbindung ins Internet gewährleistet. In einem weiteren Schritt soll erreicht werden, dass Datenströme auf ein neues Netzwerk übertragen werden können.

3.2 Aufgabe

Die Aufgabe soll in folgenden Teilschritten gelöst werden:

- Gründliche Analyse der Problemstellung
- Entwicklung einer Testumgebung zur Simulation der Problemstellung
- Entwicklung einer Applikation, die sich laufend mit offenen Netzwerken verbindet.
- Eigenentwicklung oder Implementation eines Roaming-Protokolls, welches zumindest das Lesen von Email-Nachrichten erlaubt.

3.2.1 Randbedingungen und Infrastruktur

- Die Applikation soll unter Linux laufen
- Die Applikation soll unter einer Open-Source Lizenz auf sourceforge.net veröffentlicht werden.

Kapitel 4

Grundlagen

4.1 Wireless Lan 802.11

Der IEEE Standard 802.11 spezifiziert den Mediumzugriff (MAC-Layer) und die physikalische Schicht für drahtlose Netzwerke. Es gibt diverse Erweiterungen für den 1997 verabschiedeten Standard. Diese unterscheiden sich hauptsächlich in ihrem verwendeten Frequenzband und in der Datenübertragungsrate.

Standard	Frequenzband	Datentransfer	Verabschiedet
802.11	2,400 bis 2,485 GHz	2 MBit/s	1997
802.11a	5 GHz	54 MBit/s	1999
802.11b	2,400 bis 2,485 GHz	11 MBit/s	1999
802.11g	2,400 bis 2,485 GHz	54 MBit/s	2003
802.11n	5 GHz (geplant)	540 MBit/s	Ende 2006 (geplant)
802.11p	5.850-5.925 GHz (geplant)	27 MBit/s	Ende 2008 (geplant)

Abbildung 4.1: 802.11 Standards

Am weitesten verbreitet sind 802.11b (11 MBit/s) und 802.11g (54 MBit/s). Die Frequenzen im 2.4GHz band 14 Kanäle aufgeteilt, von denen in Europa aber nur 13, in den USA nur 11 benutzt werden dürfen.

Die Verbindung von Wirelessgeräten erfolgt entweder im Ad-Hoc-Modus, bei welchem alle Geräte direkt miteinander Kommunizieren, oder über den Infrastruktur-Modus, bei welchem die Kommunikation über eine Basisstation (Accesspoint) erfolgt. Diese ist meist auch gleich die "Bridgeäuf das restliche Netzwerk (Distribution System, DS). Meistens wird der Infrastruktur-Modus eingesetzt.
Jeder Accesspoint definiert eine Zelle, genannt Basic Service Set(BSS). Mehrere Zellen können zu einem logischen Netzwerk zusammengeschlossen werden. In diesem Fall spricht man von einem Extended Service Set(ESS).

Damit eine Wirelesskarte sich mit einem solchen Netzwerk verbinden kann,

Kanal	Frequenz
1	2.412
2	2.417
3	2.422
4	2.427
5	2.432
6	2.437
7	2.442
8	2.447
9	2.452
10	2.457
11	2.462
12	2.467
13	2.472
14	2.484

Abbildung 4.2: 802.11 Frequenzen

benötigt sie also verschiedene Informationen: Die ESSID(Netzwerkname) des logischen Netzwerks, die BSSID(Zell-ID, Mac-Adresse des Accesspoints) und den Kanal. Mittels sog. Beacon-Frames werden diese Informationen periodisch gesendet, so können sich auch Stationen verbinden, die bisher keine Informationen über das betreffende Netzwerk hatten. Aus Sicherheitsgründen kann diese Funktion auch ausgeschaltet werden.

Medienzugriff

Die Art und Weise, wie Daten über eine Wirelessverbindung verschickt werden ist klar geregelt. Standardmässig sind auf einem Wirelessnetz mehrere Clients. Es muss sichergestellt werden, dass alle Stationen mit dem Accesspoint kommunizieren können, ohne sich gegenseitig zu beeinflussen. Standardmässig geschieht dies über CSMA/CA. Dabei prüft eine Station erst, ob das Medium frei ist, indem sie es abhört. Ist das Medium für die Dauer eines sogenannten Interframes frei, wird dieses gesendet. Sollte das Medium belegt sein, wartet die Station eine zufällige Zeit lang und versucht danach erneut das Frame zu senden. Bei jedem Frame, das über das Medium übermittelt wird, beschreibt das Time-Feld, wie lange es dauern wird, dieses Frame zu übermitteln. Sendewillige Stationen, welche das Medium abhören, können somit abschätzen, wann das Medium wieder frei sein wird.
Damit sichergestellt wird, dass alle Daten richtig übertragen werden, wird jedes von einer Station erhaltene Frame mit einem ACK quittiert (Broadcast-Pakete werden nicht quittiert). Tritt während der Übermittlung ein Fehler auf, wird das Paket automatisch erneut übermittelt.

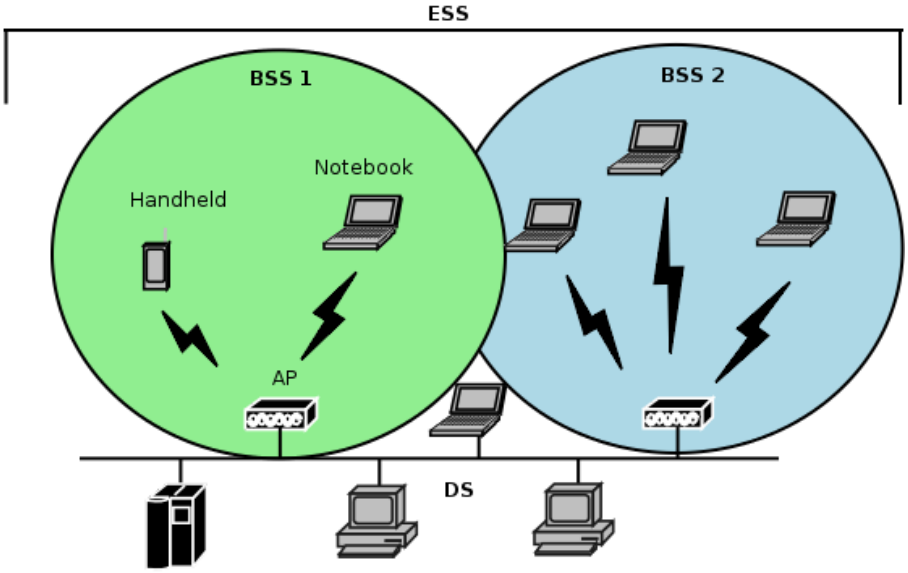


Abbildung 4.3: Wireless Basics

4.2 Wardriving

Unter Wardriving versteht man das systematische Suchen von Wireless-Netzwerken mithilfe eines Fahrzeugs (Auto, Bus etc). Ausgerüstet mit einem Notebook oder PDA, einem Sniffer und evtl. einem GPS Gerät spürt der Wardriver so möglichst viele offene(unverschlüsselte) Netze auf, um über diese gratis im Internet zu surfen.

Teil II

Systementwicklung

Kapitel 5

Leistungsmerkmale

Vor allem in dicht besiedelten Gebieten ist die Konzentration von offenen Wireless-Netzwerken mittlerweile so hoch, dass man versuchen könnte, Daten auch mobil, d.h. z.B. während einer Busfahrt zu transferieren. Das Problem hierbei ist, dass die Zeit, in der man sich in der Nähe eines Accesspoints befindet meist nicht reicht, um eine manuelle Verbindung herzustellen, geschweige denn ganze Email-Nachrichten herunterzuladen.
Es ist nun eine Software zu entwickeln, welche die Suche und den Verbindungsaufbau zu einem Accesspoint automatisiert. Dadurch soll es in dichtbesiedelten Gebieten möglich sein eine quasi stetige Verbindung zu simulieren, über welche Daten transferiert werden können, auch wenn die physische Verbindung dabei öfters auf- und abgebaut werden muss.
Konkret sind die folgende Funktionalitäten zu implementieren:

- Erkennung
  Die aktiven Wireless Netzwerke in der näheren Umgebung sollen erkannt und aufgelistet werden. Dabei sollen gewisse, für den weiteren Verlauf wichtige, Daten erfasst und für den weitere Verarbeitung zur Verfügung gestellt werden.

- Verbindungsmanagement
  Mittels eines geeigneten Algorithmus soll das beste Netz herausgefiltert und versucht werden die Einstellungen derart zu verändern, dass es möglich ist, sich mit diesem Netz zu verbinden.
  Ist eine Verbindung einmal hergestellt, soll die Qualität dieser Verbindung permanent überwacht werden und sobald die Verbindunsqualität unter einen gewissen Schwellwert sinkt, eine neue Verbindung gesucht werden.

- Sicherstellen der Erreichbarkeit
  Der ständige Wechsel der Netze, mit denen der Warsurfer sich während einer Session verbindet, hat stetige Unterbrüche der Verbindung ins Internet zur Folge. Es soll eine Möglichkeit gefunden werden, eine Verbindung virtuell über verschiedene Netze aufrecht zu halten.

- Überwachen der Verbindung
  Sobald Einstellungen für ein gewisses Netz gemacht worden sind, sind diese auch zu verifizieren und gegebenenfalls anzupassen oder durch andere zu

ersetzen. Dabei muss beachtet werden, dass die Zeit in welcher sich der Client auf einem aktiven Netz befindet, kurz ist, sodass ein DHCP-Request nur im Falle einer längeren Verbindung mit dem Netzwerk erfolgreich ist.

- Sicherheit
  Da auch vertrauliche Daten wie Emails oder Passwörter über fremde Netzwerke verschickt werden, ist ein geeigneter Mechanismus zu implementieren, mit dem die übermittelten Daten gegen fremden Zugriff gesichert sind.

- Statistik
  Alle Daten, die während eines Rides erfasst werden, sollen derart zur Verfügung gestellt werden, dass es mit geringem Aufwand möglich ist, von den gesammelten Daten Statistiken zu erstellen. Dabei sollen vor allem folgende Eigenschaften festgehalten werden:

  ssid
  Es soll sowohl die ESSID als auch die BSSID eines Netzes zur Verfügung gestellt werden.

  IP Einstellungen
  Es soll sichergestellt werden, dass ohne weiteres auf ein bereits funktionierendes Netz erneut zugegriffen werden kann. Somit sind sämtliche Daten, wie IP-Adresse, Route, Standardgateway und Nameserver zu speichern.

  Netzverfügbarkeit
  Es soll gespeichert werden, wann ein Netz das erste mal gesichtet worden ist, wie lange es während den unterschiedlichen Rides verfügbar war.

- Selbstlernend
  Die Software soll lernend sein. Netzwerke, die schon einmal erfasst worden sind, sollen wieder erkannt werden. Dies soll helfen an einem Netz unterschiedliche Tests sukzessive durchführen zu können, da die Zeit oft nicht ausreicht um alle Tests auf ein Netz während eines Rides durchzuführen.

- Erweiterbarkeit
  Es wird Wert auf die Erweiterbarkeit gelegt. Es soll möglich sein die Software beispielsweise durch ein GUI oder verschiedene Plugins zu bereichern und deren Funktionsumfang zu erweitern.

- Konfigurierbar Dem Anwender soll die Möglichkeit gegeben werden eigene Konfigurationen zu erstellen. Eine Konfiguration soll mindestens folgende Daten erfassen:

  - IP Adresse
  - ssid
  - WEP
  - MAC Adresse
  - Standard Gateway
  - DNS

Die Software wird ausschliesslich mit Open-Source Software erstellt. Der Quellcode wird, sobald die erste Version verfügbar sein wird auf sourceforge gestellt und somit der Welt zugänglich gemacht werden. Die Zielplattform ist Debian GNU Linux, wobei auch andere Linux-Distributionen unterstützt werden sollen. Zusätzlich zur Software wird eine Benutzerdokumentation in deutscher und in englischer Sprache erstellt.

Kapitel 6

Analyse

6.1 Information Gathering

Um sich mit einem Netz verbinden zu können, sind Informationen über ein Netzwerk unabdingbar. Die MAC Adresse des Accesspoints, auch bssid genannt, oder der Name, des Wireless-Netzes müssen zwingend vorhanden sein. Wichtig ist auch zu wissen, ob das Netz noch aktiv ist oder nicht. Oder ob auf dem Netz noch mit Daten zu rechnen sind, oder der Empfang bereits so schwach ist, dass kaum mehr Datenverkehr zwischen einem Client und dem Accesspoint herrscht. Es gibt verschiedene Wireless Network-Sniffer mit unterschiedlichem Einsatzgebiet. Einige beschränken sich in der Funktion darauf bloss Drahtlose Netzwerke aufzuspüren, während andere versuchen weitere Informationen über ein Netz zu sammeln oder gar schwache WEP-Schlüssel zu knacken.

Wichtige Kriterien für die Auswahl eines Sniffers sind:

- Angabe der ESSID
- Angabe der BSSID
- Angabe eines möglichen IP-Ranges
- Angabe ob ein Netz offen oder verschlüsselt ist
- Komfortable Bereitstellung der Informationen, sodass ein Output des Programms gut weiter verwendet werden kann.

Airsnort
Airsnort[1] ist ein in der Wardriver-Szene weit verbreitetes Tool zum auffinden von Wireless Netzwerken. Es ist sticht vor allem durch seine einfache Bedienung heraus. Ebenfalls erwähnenswert ist die Eigenschaft, dass dieses Tool über die Möglichkeit verfügt, WEP-Keys zu entschlüsseln.
Für unsere Ansprüche kommt Airsnort nicht in Frage, da man die Daten nicht in Echtzeit abfragen kann. Airsnort bietet zwar die Möglichkeit in ein File zu loggen, was aber bei der getesteten Version nicht funktioniert hat.

Name	Homepage	GUI	Server	Feile	WEP Dec	ESSID	BSSID	Version
Airsnort	airsnort.shmoo.com	Y	N	Y	N	Y	Y	0.2.7
Ethereal	www.ethereal.com	Y	N	N	N	Y	Y	0.10.12
WifiRadar	www.bitbuilder.com	Y	N	N	N	Y	Y	0.1.9.4
WlanScanner	www.cweiske.de	Y	N	N	N	Y	Y	0.2
Kismet	www.kismetwireless.net	Y	Y	Y	N	Y	Y	2004.04.R1

Abbildung 6.1: Vergleich der Wireless Sniffer

Ethereal
Ethereal[2] ist ein Sniffer, der sich weit verbreitet hat. Er hat seine Stärken vor allem im Netzwerkbereich wenn es darum geht den Verkehr in einem Netzwerk zu überwachen. Mit der Möglichkeit Filter zu setzen, kann der Netzwerkverkehr paketgenau analysiert werden. Ethereal kommt mit einem praktischen GUI daher und bietet eine Vielzahl an Konfigurationsmöglichkeiten.
Für einen Einsatz als Wireless-Sniffer macht sich Ethereal eher weniger gut. Ethereal zeigt jedes einzelne Paket an, das er empfangen hat. Dies bedeutet einen enormen Aufwand wenn man daraus verwertbare Daten über verfügbare Drahtlosnetzwerke ableiten muss. Ausserdem biete Ethereal nicht die Möglichkeit die Daten in Echtzeit einem anderen Programm zur Verfügung zu stellen. Somit scheidet er als möglicher Sniffer aus.

WifiRadar
WifiRadar[3] ist ein kleines Tool, welches dafür gebaut worden ist einem einzelnen Anwender die Konfiguration seiner Wirelesskarte zu erleichtern. Er wurde hauptsächlich deshalb untersucht, weil er ebenfalls eine Scan-Funktionalität bietet und durch seine Kompaktheit besticht.
WifiRadar scannt als erstes nach verfügbaren Netzen. Fr jedes Netz kann der User ein Profil eingeben, welches die Netzwerkeinstellungen beinhaltet. Kommt nun ein bekanntes Netz in Reichweite, konfiguriert WifiRadar die Netzwerkkarte entsprechend dem vorhandenen Profil.
Fr uns scheidet WifiRadar aus, weil er die Information über verfügbare Netze nur über ein GUI anbietet.

WlanScanner
WlanScanner[4] ist laut Autorangaben das Tool für Linux, was Netstumbler für Windows ist. Er scannt nach Netzen und listet diese in einem User-Interface auf. Dabei kann angezeigt werden, ob ein Netz verschlüsselt ist oder nicht.
Für uns kommt WlanScanner nicht in Frage, weil er einerseits die Information über die gefundenen Netze nur über sein User-Interface zur Verfügung stellt. Andererseits ist dieses Tool in PHP geschrieben. PHP für diesen Zweck zu vergewaltigen entspricht nicht unserer Programmierphilosophie, was ebenfalls ein wichtiger Grund für den Ausschluss von WlanScanner war.

Kismet
Kismet[5] ist ein 802.11 Layer 2 WLAN Sniffer und ein Intrusion Detection System in einem. Kismet funktioniert mit jeder Wirelesskarte, welche den Monitor-

Modus unterstützt. Kismet ist der State-of-the-art in Sachen wardriving. Dies ist nicht zuletzt dem Umstand zu verdanken, dass der Output und die Logfiles von Kismet ohne Modifikation mit anderen Programmen (Bsp. Airsnort, Ethereal) weiterverarbeitet werden können. Eine der grössten Stärken von Kismet ist auch die Client/Server Architektur, welche einen einfachen Weg darstellt in Echtzeit an die aktuellen Daten von Kismet zu gelangen. Kismet bietet eine Telnet-Schnittstelle an, über die genau spezifiziert werden kann, welche Informationen übermittelt werden sollen.
Da der Funktionsumfang und die Weiterverarbeitbarkeit der Daten alle anderen Programme in den Schatten stellt, fällt der Entscheid nicht schwer. Kismet wird als Sniffer im Warsurfer eingesetzt.

6.2 Kenngrössen der Netzwerkerkennung

Wireless Netzwerke lassen nur erkennen, wenn auch Daten versandt werden, die von einem Sniffer mitgehört werden können. Dies ist beispielsweise der Fall, wenn ein Client gerade auf diesem Netz surft. Dieses Szenario stellt aber einen Idealfall dar, von dem nicht ausgegangen werden kann. Aber auch wenn nicht gerade ein Client am surfen ist, gibt es Pakete, die die Präsenz eines Drahtlosnetzwerkes anzeigen. Beacons sind Frames, die ein Accesspoint periodisch aussendet. Sie beinhalten für Clients wichtige Informationen. So steht in einem Beacon die aktuelle Zeit, die SSID des Accesspoints, sowie ob WEP-Verschlüsselung aktiviert ist oder nicht. Fr die Identifikation eines Netzwerkes dient der SSID (**S**ervice **S**et **ID**entifier), durch welchen ein Client erfährt mit welchem Accesspoint er sich verbinden kann.
Ein herkömmliches Beacon-Frame ist ungefähr fünfzig Bytes lang, wovon die Hälfte für Header und CRC verbraucht wird. Der Header enthält ausserdem Source- und Destination MAC Adressen, sowie weitere für den Kommunikationsprozess wichtige Informationen. Die Destination MAC Adresse in einem Beacon-Frame ist immer die Broadcast MAC, so dass sichergestellt ist, dass alle Clients dieses Frame auch erhalten. Des weiteren sind aber folgende Felder in einem Beacon enthalten:

- Beacon Interval
  Dieser Wert bestimmt das Zeitintervall zwischen zwei Beacon-Frames. Dies ist für eine Station, die im Powersave-Mode arbeitet wichtig um zu wissen, wann sie wieder aufwachen und auf ankommende Pakete horchen soll.

- Timestamp
  Nachdem eine Station ein Beacon-Frame erhalten hat, verwendet sie den Timestamp um die Systemuhr einzustellen. Dies ermöglicht die Synchronisation aller Clients, die auf diesen Accesspoint verbunden sind.

- **S**ervice **S**et **ID**entifier
  Der SSID identifiziert ein Wireless Netzwerk eindeutig. Normalerweise wird die SSID in jedem Beacon Frame mitgeschickt. Dies ermöglicht es einem Client, ein Drahtloses Netzwerk zu erkennen. (Windows XP nutzt diese Eigenschaft aus um den User über verfügbare Drahtlosnetzwerke zu

informieren) Um die Sicherheit ein wenig zu erhöhen bieten verschiedene Anbieter von Accesspoints aber an, den SSID-Broadcast zu unterbinden.

- Supported Rates
  Durch dieses Feld teilt der Accesspoint seinen Clients mit, mit welcher Geschwindigkeit Daten übermittelt werden können. Sollte ein Accesspoint nur 1, 2 und 5.5mbps unterstützen, könnte ein Client nicht mit 11mbps Datenrate übertragen und deshalb eventuell einen anderen Accesspoint bevorzugen.

- Parameter Sets
  In diesem Feld werden dem Client weitere Informationen über die Signalisierung gegeben. (Beispielsweise Hopping Schema, Frequency-Spreading etc.)

- Capability Information
  In diesem Feld wird mitgeteilt, ob der Accesspoint WEP verwendet, oder ob es sich um ein offenes Netz handelt.

- Traffic Indication Map
  Ein Accesspoint benutzt dieses Feld um einem Client, welcher sich im PowerSave-Modus befindet, mitzuteilen, dass sich Traffic für diesen Client im Puffer des Accesspoint befindet.

Die meisten in einem Beacon übermittelten Daten muss sich der Anwender keine Gedanken machen, weil sie vom Treiber der Karte verarbeitet werden. Die Informationen die verarbeitet werden müssen sind ESSID, BSSID sowie ob WEP-Verschlüsselung aktiviert ist oder nicht.

6.3 Assoziation

Bevor Daten über eine Drahtlose Netzwerkverbindung verschickt werden können, muss sich eine Wireless Station erst mit einem Accesspoint verbinden. Die Verbindung geschieht anhand eines Zustandsautomaten, welcher die Art der Pakete, die von einer Wireless Station aus verschickt werden dürfen, regelt.
Wie auf Abb. 6.2 zu erkennen ist, befindet sich eine Wireless Station nach dem Aufstarten in einem Zustand in welchem sie weder authentifiziert noch assoziiert ist[6]. Die Karte wechselt in einen Scanmodus um zu erfahren ob ein Accesspoint in der Nähe ist. Dies kann auf zwei unterschiedliche Arten geschehen. Im passiven- oder im aktiven Modus.
Befindet sich die Karte im passiven Modus, horcht sie lediglich auf Beacon Frames um einen Accesspoint zu identifizieren. Im aktiven Modus verschickt die Station sogenannte Probe Frames, welche von einem Accesspoint beantwortet werden. Befinden sich mehrere Accesspoints im Empfangsradius der Station, werden diese anhand ihrer Signalstärke und der Fehlerrate sortiert und die Station versucht sich mit dem besten Accesspoint zu verbinden.

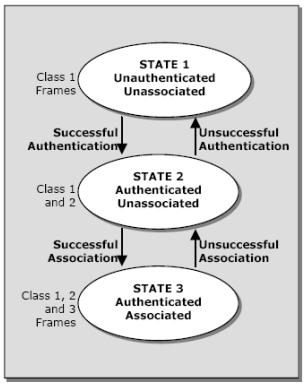


Abbildung 6.2: Die verschiedenen Assoziationsstati

Um sich bei einem Accesspoint zu authentisieren, schickt die Station ein Authentication-Request-Frame[7] and den Accesspoint. Dieser Antwortet darauf mit einem Authentication Response-Frame. Nun ist die Station zwar authentisiert, aber noch nicht mit dem Accesspoint assoziiert. Die Station sendet nun ein Association Request-Frame an den Accesspoint, welcher dieses Paket mit einem Association Response-Frame beantwortet. Erst jetzt ist die Station sowohl authentifiziert als auch assoziiert. Die Grundvoraussetzung für einen erfolgreichen Datenaustausch mit dem Accesspoint und somit dem Internet ist nun gegeben.

802.11 Frames

Die Frames, die zwischen Accesspoint und Wireless Station verschickt werden, entsprechen einem bestimmten Format[8], welches hier etwas genauer untersucht und erläutert werden soll.

Frame Control	Duration	Address 1	Address 2	Address 3	Seq	Address 4	Data	Checksum
2	2	6	6	6	2	6	0-2312	4

Die Frame Control Struktur im Detail:

Version	Type	Subtype	To DS	From DS	MF	Retry	Pwr	More	W	O
2	2	4	1	1	1	1	1	1	1	1

Abbildung 6.3: 802.11 Frames detailliert

- **Duration**
  Dieser Wert wird benutzt um den Netzwerk Allocation Vector zu berechnen.

- **Address 1-4**
  Ein Frame kann bis zu 4 Source- oder Destinationadressen enthalten. (Gemäss “From DS” oder “To DS” Feld)
- **Sequence Control**
  Dieses Feld enthält die Sequenz- oder Fragmentnummer, welche verwendet werden um Frames in der richtigen Reihenfolge zusammenbauen und duplizierte Frames erkennen zu können.
- **Data**
  Daten, die mit diesem Frame transportiert werden.
- **CRC**
  32-Bit Cyclic Redundancy Check
- **Version**
  Bezeichnet die Version des 802.11 Standards.
- **Type**
  Bezeichnet den Frame Typ. Mögliche Werte sind: Management, Control und Data
- **Subtype**
  Bezeichnet den Subtyp. Mögliche Werte sind: Authentication frame, Deauthentication frame; Association request frame; Association response frame; Reassociation request frame; Reassociation response frame; Disassociation frame; Beacon frame; Probe frame; Probe request frame and Probe response frame.
- **To DS**
  Dieser Wert ist auf 1 gesetzt, wenn das Paket an ein Disribution System gesendet wird.
- **From DS**
  Dieser Wert ist auf 1 gesetzt, wenn das Paket von einem Dustribution System empfangen wird.
- **MF**
  Das More Fragment Bit ist gesetzt, wenn diesem Frame-Fragment noch weitere Fragmente folgen werden.
- **Retry**
  Das Retry Bit ist dann gesetzt, wenn ein Fragment eines Frames erneut angefordert wird.
- **Pwr**
  Bezeichnet den Powermanagement-Modus in welchen sie die Station nach der Übermittlung begeben wird.
- **More**
  Zeigt an, dass sich im Puffer für diese Station noch mehrere Frames befinden.
- **W**
  Zeigt an, dass der Body des Frames WEP-verschlüsselt ist.

- **O**
  Order zeigt an, dass das Frame gemäss einer bestimmten Order-Class verschickt worden ist.

6.4 Default settings

Von Accesspoint zu Accesspoint können Einstellungen, wie der Standardgateway oder der Namensserver sowie der IP-Range variieren. Normalerweise stellt dies kein Problem dar, weil alle diese Informationen durch einen DHCP-Server, welcher in den meisten Netzen aktiv ist, zur Verfügung gestellt werden. Während einer Busfahrt reicht die Zeit aber nicht aus um für jedes Netz einen DHCP-Request abzusetzen. Der einzig richtige Ansatz an dieser Stelle ist zu versuchen diese Einstellungen zu erraten und auszuprobieren.
Für den Warsurfer kommen lediglich solche Netze in Frage, welche nicht verschlüsselt sind. Es sei denn, es liegt eine bekannte Netzwerkkonfiguration vor, welche durch den Benutzer eingefügt worden ist.
Um die Netzwerkeinstellungen zu erraten haltet sich der Warsurfer an gewisse Fakten und versucht anhand derer eine funktionierende Verbindung herzustellen.

- Kismet Daten
  Es kommt vor, dass es Kismet gelingt den IP-Adress-Range eines Netzes mitzusniffen. Ist nun eine IP-Adresse aus einem Netz bekannt, wird eine /24 Netzmaske angenommen. Des weiteren wird angenommen, dass der Standardgateway eine IP-Adresse mit der Endung .1 hat. Anhand dieser Daten ist es nun möglich eine IP-Adresse, sowie eine Route ins Internet zu setzen.

- Standard ESSID
  Wenn ein Accesspoint direkt vom Hersteller kommt, ist eine Standard ESSID eingestellt. Wird eine solche von Kismet mitgesnifft, wird angenommen, dass der Betreiber des Accesspoints diesen nicht weiter konfiguriert hat. Somit kann von der ESSID ausgehend auf die IP-Adresse, sowie auf die Standard-Route bzw. den Standardgateway geschlossen werden.

- Hersteller MAC (BSSID)
  Jeder Accesspoint ist mit einer weltweit eindeutigen MAC-Adresse ausgestattet. Die ersten 3 Bytes dieser Adresse sind vorgegeben und identifizieren den Gerätehersteller. Die Einstellungen, die nun vorgenommen werden beruhen auf der Annahme, dass ein Betreiber eines Accesspoints, der die MAC-Adresse des Accesspoints nicht ändert und diesen nicht verschlüsselt, wohl auch nicht die IP-Einstellungen verändert. Der Betreiber wird wohl die Standard-Einstellungen beibehalten und im Sinne der Personalisierung seines Netzes nur den Netznamen verändern. Sollte dies der Fall sein, ist es auch möglich mit dem Accesspoint zu verbinden, selbst wenn der Betreiber SSID Broadcast ausgeschaltet haben sollte. Um sich mit einem Accesspoint zu assoziieren reicht es aus, die BSSID anzugeben.

- Standard IP Adressen
  Erfahrungsgemäss ist der IP-Range, der am meisten vorkommt 192.168.1.0/24. Der Standardgateway ist 192.168.1.1. An diesen werden auch alle Anfragen, wie beispielsweise Nameserver-Anfragen geschickt und von diesem bearbeitet. Sollte dies nicht funktionieren, wird alternativ ein 192.168.0.0/24-er Netz eingestellt. Die Annahmen betreffend Gateway und Nameserver bleiben sich aber gleich.

Sollte es nach all diesen Massnahmen nicht klappen eine Verbindung ins Internet herzustellen, wird versucht per DHCP an die richtigen Einstellungen zu gelangen. Dies dauert in der Regel länger als ein Netz verfügbar ist und wird deshalb als letzte Möglichkeit in betracht gezogen.
Sollte einmal eine Konfiguration funktioniert haben, wird diese gespeichert und bei einem neuerlichen Lauf eingestellt.

6.5 Netzverfügbarkeit

Es ist wichtig seine Umwelt zu kennen. Aus diesem Grund muss in etwa abgeschätzt werden können, wie lange ein Netz verfügbar ist. Der Warsurfer soll in Tram und Bus zum Einsatz kommen. Unzählige Messungen in diesem Bereich haben ergeben, dass ein Netz im Schnitt während 5 Sekunden sichtbar ist.

6.6 Rechtliche Einschränkungen

Wardriving bewegt sich in einer rechtlichen Grauzone[9]. Wardriving ist das Eindringen in ungeschützte, drahtlose Netzwerke durch private Personen. Für die rechtliche Beurteilung sind 3 Aspekte wichtig.

1. Aktivitäten der Wardriver
2. Ziele der Wardriver
3. Verwendete Software

Auch beim Wardriving gilt der Grundsatz: “Kein Angeklagter ohne Kläger”. Damit der Kläger seinerseits aber Klagen kann, muss eine gesetzliche Grundlage gegeben sein. Beim Wardriving sind dies vor allem die folgenden:

- StGB 147:
  Betrügerischer Missbrauch einer Datenverarbeitungsanlage
- StGB 251:
  Urkundenfälschung
- StGB 143:
  Unbefugte Datenbeschaffung
- StGB 143bis:
  Unbefugtes Eindringen in ein Datenverarbeitungssystem
- StGB 144bis:
  Datenbeschädigung

Betrügerischer Missbrauch einer Datenverarbeitungsanlage

Damit dieser Tatbestand erfüllt ist, müssen folgende Elemente vorliegen:

- unrichtige, unvollständige oder unbefugte Verwendung von Daten
- Herbeiführen von Vermögensverschiebung zum Schaden einer Drittperson
- Absicht der unrechtmässigen Bereicherung

All die genannten Elemente treffen für den Warsurfer nicht zu. Es werden keinerlei Daten dritter verändert oder eingesehen. Es wird keine Vermögensverschiebung vorgenommen und die Einzige Bereichung, die der Benutzer erfährt, ist ein paar Sekunden lang in den Genuss einer Internetverbindung zu kommen, was aber im rechtlichen Sinne kaum als Bereicherung aufgefasst werden kann.

Urkundenfälschung

Damit der Tatbestand der Urkundenfälschung vorliegt, müssen folgende Elemente zutreffen:

- Fälschung von Urkunden gemäss StGB 110 Ziff. 5
- Absicht der Vermögensschädigung bei einer Drittperson
- Absicht des unrechtmässigen Vorteils

Urkundenfälschung ist beim Warsurfen kein Thema, da keinerlei Dokumente des Netzbetreibers eingesehen werden.

Unbefugte Datenbeschaffung

Damit der Tatbestand der Urkundenfälschung erfüllt ist, müssen folgende Elemente zutreffen:

- Beschaffung von gespeicherten oder übermittelten Daten
- Daten sind nicht für den Wardriver bestimmt (unbefugte Datenverwendung)
- Daten sind gegen unbefugten Zugriff besonders geschützt
- Absicht der unrechtmässigen Bereicherung

Da mit dem Warsurfer standardmässig nur auf Netze zugegriffen wird, welche nicht besonders geschützt sind und der Benutzer nicht die Absicht hat sich unrechtmässig zu bereichern, dürfte auch hier der Tatbestand nicht erfüllt sein.

Unbefugtes Eindringen in eine Datenverarbeitungsanlage

Damit der Tatbestand des Unbefugten Eindringens in eine Datenverarbeitungsanlage erfüllt ist, müssen folgende Elemente zutreffen:

- Eindringen in ein fremdes Datensystem
- Daten sind nicht für Wardriver bestimmt (unbefugte Datenverwendung)
- Daten sind gegen unbefugten Zugriff besonders geschützt

Dieser Tatbestand ist beim Warsurfer ebenfalls nicht erfüllt, da standardmässig nur auf Netze zugegriffen wird, die nicht verschlüsselt und somit nicht gegen unbefugten Zugriff geschützt sind.

Datenbeschädigung

Damit der Tatbestand der Datenbeschädigung erfüllt ist, müssen folgende Elemente zutreffen:

- Veränderung, Löschung oder Unbrauchbarmachen von gespeicherten oder übermittelten Daten
- Daten sind nicht für Wardriver bestimmt (unbefugte Datenverwendung)
- Vorsatz (Fahrlässigkeit reicht für diesen Tatbestand nicht aus)

Beim Warsurfen werden keinerlei Daten verändert. Somit trifft dieser Tatbestand ebenfalls nicht zu.

Fazit

Wird der Warsurfer so benutzt, wie es von den Entwicklern gedacht ist, bestehen keinerlei rechtliche Bedenken. Es ist aber durchaus möglich den Warsurfer so zu benutzen, dass Rechtsverletzungen auftreten. Die Entwickler sprechen sich allerdings ganz klar dagegen aus!

6.7 Verbindungsunterbrüche

Sobald sich der Warsurfer einmal mit einem Netz verbunden hat, werden so lange Daten transferiert, bis das Netz nicht mehr verfügbar ist. Es gibt unterschiedliche Indikatoren dafür, ob ein Netz noch aktiv ist oder nicht.

- Link level
- Noise
- Signalstärke

Es hat sich durch überwachen der einzelnen Werte während dem wardriving herausgestellt, dass der verlässlichste dieser Werte der Link level ist. Er berechnet sich aus der Signalstärke / Noise. Obwohl dies der zuverlässigste Wert ist, kann es Schwankungen zwischen verschiedenen Drivern geben, welche diese Werte jeweils unterschiedlich angeben.

Fällt die Signalqualität eines Netzes unter einen bestimmten Schwellwert, so wird das Netz nicht mehr verwendet und ein neues gesucht.
Obwohl durch die ständige Suche nach neuen Netzen die physische Verbindung dauernd unterbrochen wird, muss ein Weg gefunden werden, wie man Daten möglichst effizient vom Internet auf den mobilen Client übertragen kann.

6.8 Kenngrössen eines Verbindungsalgorithmus

In der Theorie können verschiedene Szenarien auftreten, auf die man während einem Ride stossen kann. da die Software lernfähig sein soll, muss ein Algorithmus implementiert werden, welcher auf unterschiedliche Situationen angemessen reagieren kann. Ausschlaggebend sind für einen solchen Algorithmus folgende Kenngrössen:

- Uptime eines Netzes
  Wie lange ist das Netz sichtbar. Und wie lange befindet sich die Linkqualität eines Netzes in einem genügend guten Bereich.
- Linkqualität eines Netzes
- Bekanntheitsgrad eines Netzes
  Dabei wird berücksichtigt, ob das Netz bereits einmal gesehen worden ist und ob eventuell gar eine Verbindung ins Internet zu Stande gekommen ist. Ausserdem muss abgeklärt werden, welche Verbindungskonfigurationen ausprobiert worden sind, um sich mit dem Netz zu verbinden.
- Benutzerkonfiguration
  Dem Benutzer wird die Möglichkeit gegeben ihm bekannte Netze in eine Konfiguration einzutragen. Die Einstellungen, die der Benutzer dort vornimmt, sind ebenfalls zu berücksichtigen.

Aus diesen Daten soll der Algorithmus das am besten geeignete Netz suchen. Natürlich spielt auch eine gewisse Philosophie mit. Es stellt sich die Frage, wo man den Schwerpunkt setzen will. Will man möglichst viele Netze kennenlernen um in einem weiteren Ride die bestmögliche Verbindung zur Verfügung stellen zu können, oder soll versucht werden, mit den bisher bekannten Netzen eine Verbindung herzustellen und dafür nicht so viele Netze neu kennen zu lernen?

6.9 Roaming

Dieser Teil befasst sich mit dem Problem, dass Daten zwar während einer Session an ein und denselben Ort verschickt werden, dass sie aber nicht immer vom selben Ort herkommen. So kann es sein, dass der Client sich zum Zeitpunkt an dem er einen Request an einen Webserver verschickt die IP 80.218.39.2 hat, befindet. Der abgesetzte Request wird vom Server bearbeitet und an die Adresse 80.218.39.2 zurückgeschickt. Unterdessen hat sich der Client aber schon in das nächste Netz bewegt, und hat nun die IP 84.73.172.138. Der Client wird also niemals eine Antwort vom Server erhalten. Nun hätte er natürlich die Möglichkeit einen neuerlichen Request abzusetzen und mit etwas Glück würde irgendwann

auch einmal lange genug eine Verbindung mit einem Netz bestehen, um auch Daten empfangen zu können. Dieser Ansatz befriedigt aber nur wenig bis gar nicht. Nachfolgend werden Lösungsvarianten für das oben beschriebene Problem vorgestellt.

6.9.1 Autonomer Client

Der autonome Client stellt die einfachste Variante des Warsurfings dar. Der Client verbindet sich dabei direkt mit dem Internet, ohne den Traffic über einen Server zu schicken, welcher das Roaming über verschiedene Netze ermöglicht. Der Vorteil dieser Variante ist die Einfachheit. Der Client verbindet sich direkt mit dem Internet. Mittels http continue, könnte ein Download fortgesetzt werden, sobald der Client wieder am Internet ist. Der offensichtliche Nachteil dabei ist, dass längst nicht alle gängigen Protokolle die Wiederaufnahme abgebrochener Transfers unterstützen. Der Vorteil liegt in der einfachen Implementation. Ein Laptop mit installiertem Warsurfer reicht vollständig aus.

6.9.2 Home Proxy

Wie die Grafik zeigt, ist dieses Setup nicht ganz so einfach wie dasjenige des Autonomen Clients. Es gibt einen Home Proxy, der die eigentliche Kommunikation mit dem Webserver übernimmt und die Daten an den Client zurück schickt, sobald dieser das nächste mal ins Internet kommt.

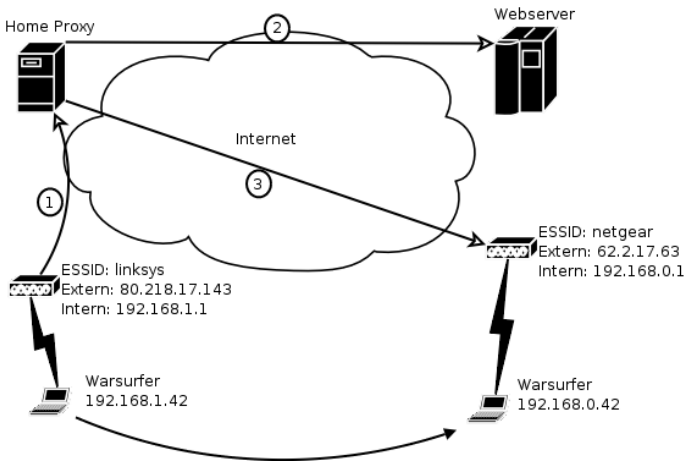


Abbildung 6.4: Funktionsweise des Home Proxy

1. Der Client schickt dem Home Proxy einen Job 2. Der Home Proxy bearbeitet diesen Job und lädt Daten von einem Server im Web 3. Sobald der Client das nächste mal im Internet ist, erhält er vom Server die verlangten Daten.
Ein solcher Ansatz hat den Vorteil, dass die Jobs, die dem Server übermittelt werden, priorisiert werden können, sodass dem Client erst einmal die wichtigeren Daten zugeschickt werden und nicht die Leitung mit weniger wichtigen

Informationen “verstopft” wird.
Um diesen Ansatz zu verwirklichen müssen alle benötigten Protokolle vereinheitlicht d.h. z.b. auf HTTP abgebildet werden. Allerdings stellt sich die Frage, ob die Implementation eines solchen Protokolls nicht den Rahmen einer Diplomarbeit sprengen würde.

6.9.3 MobileIP

Alle bisher erklärten Ansätze haben den Nachteil, das Sie nur bedingt mit dem Wechsel in ein anderes Netz umgehen können. Der Wechsel wird auf Protokollebene gehandhabt und muss so für jede Anwendung neu Implementiert werden. Erwünscht wäre eigentlich die Handhabung bereits auf dem IP-Layer, so dass jedes IP-basierte Protokoll automatisch mit dem häufigen Netzwechsel zurecht kommt, ohne das spezielle Anpassungen nötig sind. Der MobileIP[10] Standard (RFC 2002-2006) nimmt sich diesem Problem an. Dabei nimmt ein Home-Agent (ein Host, dessen IP-Adresse nicht wechselt) die Pakete für den Mobile Node entgegen und leitet sie diesem mittels IP-Encapsulation weiter. Wechselt der Mobile Node seine IP-Adresse, teilt er dies dem Home-Agent mit, damit dieser die Pakete an die neue Adresse senden kann. Antwortpakete werden vom Mobile-Node direkt an den Ziel-Host geschickt, aber mit der IP-Adresse des Home-Agents als Absenderadresse.
Die Verbreitung von MobileIP ist sehr gering. Bis anhin war die Nachfrage

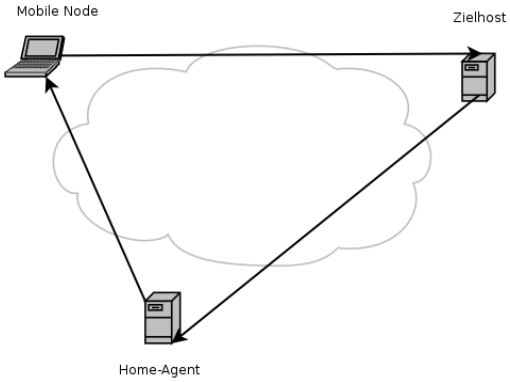


Abbildung 6.5: MobileIP

kaum da, und jetzt, wo Technologien wie Wireless langsam soweit sind, dass Mobile-Nodes (wie der Warsurfer) langsam aufkommen könnten, steht IPv6 vor der Tür, welches dieses Konzept bereits beinhaltet. Daher existieren auch keine Implementationen, die auf dem aktuellen Kernel 2.6 lauffähig wären.

6.10 Hardware

Während den ersten Tests wurde festgestellt, dass sich die verschiedenen Wirelesskarten und Treiber stark bezüglich Funktionalität und Stabilität unterscheiden. Diese Tabelle gibt eine kleine Übersicht, welche Karten für den Betrieb des Warsurfers geeignet sein könnten.

Hersteller	Modell	Anschluss	Treiber	Detector	Connector
U.S. Robotics	USR5410	PCMCIA	acx100	Funktioniert	Funktioniert
U.S. Robotics	USR805422	USB	ndiswrapper	Wird von Kismet nicht unterstützt	Ignoriert Befehle
Surecom	EP-9001	USB	zd1201	Kein Channelhopping	Funktioniert
Intel	-	Mini PCI	HostAP	Funktioniert	Funktioniert
Linksys	WPC11-v3	PCMCIA	HostAP	Funktioniert	Funktioniert
Cisco	Aironet 350	PCMCIA	airo_cs	Kein Channelhopping	Ignoriert Befehle
Lancom	AirLancer MC-54ag	PCMCIA	madwifi-ng	Wird von Kismet nicht unterstützt	Ignoriert Befehle

Abbildung 6.6: Verwendete Hardware und Treiber

Kapitel 7

Design

7.1 Testumgebung

Um die Entwicklung des Warsurfers möglichst effizient zu gestalten, ist es unumgänglich eine Testumgebung aufzubauen, mit welcher verschiedene Szenarien simuliert werden können. Dabei muss vor allem sicher gestellt werden, dass verschiedene Netze simuliert werden können.

7.1.1 OpenWRT

OpenWrt[11] ist ein Projekt, welches es ermöglicht, eine Linux-Firmware auf einem Linksys WRT54g Router von Linksys zu installieren. Dies bringt den grossen Vorteil mit sich, dass man per SSH auf diesen Accesspoint verbinden kann um dort die Netze per Script zu simulieren.

7.1.2 HostAP

HostAP ist ein Linux-Treiber für Wireless Karten mit dem Prism2 Chipset von Intersil. Mit diesem Treiber ist es möglich die Karte in den Master-Modus zu versetzen. In diesem Modus agiert der Computer, an welchem die Karte angeschlossen ist, als Accesspoint. Mit iwconfig und ifconfig können wiederum verschiedene Netze simuliert werden.

7.1.3 Scripting Engine

Damit sich nun diese Netzwerke automatisiert simulieren lassen, wurde ein Python-Interface definiert, welches eine einheitliche API auf alle verfügbaren Accesspoints darstellen soll. Das Interface wurde von uns für OpenWRT und HostAP implementiert. Aber auch Accesspoints, welche sich nicht über die Kommandozeile steuern lassen, können mit diesem Interface relativ leicht in die Testumgebung mit einbezogen werden.

Um die Engine auf einen neuen Accesspoint zu erweitern, muss von der Klasse **aphandler** abgeleitet werden und es sollten alle der folgenden Funktionen überschrieben werden, welche vom jeweiligen Accesspoint unterstützt werden:

```
class aphandler
 |  interface for ap handler
 |
 |  Methods defined here:
 |
 |  disable(self)
 |      disables wifi interface
 |
 |  enable(self)
 |      enables wifi interface
 |
 |  getdhcptable(self)
 |      returns a tuple (mac, ip, hostname)
 |
 |  login(self, hostname, user, password)
 |      log in to the device
 |
 |  logout(self)
 |      log out from the device
 |
 |  setbssid(self, bssid)
 |      sets bssid
 |
 |  setchannel(self, channel)
 |      set ap to specified channel
 |
 |  setessid(self, essid)
 |      sets essid
 |
 |  setpower(self, power=0)
 |      sets transmission power, full power if 0
 |
 |  setwepkey(self, key=None)
 |      sets wep key, or disables it if None
```

Die Scripting-Engine bietet zudem einige Hilfsfunktionen:

```
comeAndGo(aphandler, peek, secs)
    tries to simulate a 'drive-by' by increasing and decreasing the tx-power

generateRandomESSID()
    generates a random string that can be used as ESSID

generateRandomMac()
    Creates a Random Mac addresss

getProtectedPage(url, username, password, top_level_url)
```

```
retreives a password protected (basic auth) web site
```

7.1.4 Simulation

Um eine Busfahrt möglichst realitätsgetreu zu simulieren, werden ESSID, BSSID und MAC auf dem Accesspoint periodisch verändert. Dabei wird bei der ESSID eine Namenskonvention eingehalten. Dadurch werden eventuell während den Rides erstellte Statistiken nicht verfälscht, weil die simulierten Netzwerke durch ihren Namen identifizierbar sind.
Die Simulation kennt zwei verschiedene Modi. Beim einen wird eine Fahrt durch ein unbekanntes Gebiet simuliert und zufällig immer neue Netze generiert. Beim zweiten Modus wird die Lernfähigkeit des Warsurfers getestet. Dabei wird eine bestimmte Anzahl zufälliger Netze generiert, die in einer Schleife immer wieder aktiviert werden. Sollte es dem Warsurfer nun gelingen einmal eine Verbindung mit einem Netz aufzubauen, sollte er beim nächsten mal, bei dem dieses Netz entdeckt wird, als erstes die funktionierende Konfiguration austesten.

7.1.5 Feldtest

Trotz allen raffinierten Methoden, die Welt möglichst realitätsgetreu zu simulieren, sind echte Feldtests unabdingbar. Der Warsurfer muss unter anderem auch in einem realen Umfeld in Bus und Tram getestet werden. Als Einsatzgebiet ist die Limmatstadt am besten geeignet, da sie mit einer Vielzahl an Bus- und Tramlinien ein grosses Potential an offenen Netzen besitzt.

7.2 Applikationsdesign

Die Applikation wird in mehrere, voneinander unabhängige Teile getrennt. Dies bietet diverse Vorteile:

- Grössere Sicherheit, da z.b. nicht mehr der ganze Warsurfer mit root-Rechten laufen muss.
- Einfachere Entwicklung, da an jedem Teil gearbeitet werden kann, ohne das die anderen Teile direkt tangiert werden
- Flexiblere Anwendung, der Benutzer kann z.b. den Warsurfer bei einer Fahrt nur "lernen"lassen, auch wenn er keine Internetverbindung wünscht

7.2.1 Interprozesskommunikation

Die Kommunikation der einzelnen Komponenten geschieht über eine Datenbank. Dieser etwas unübliche Ansatz wird folgendermassen begründet:

- Schnell implementiert, es muss kein eigenes Protokoll definiert werden
- Die Kommunikation ist automatisch asynchron und somit flexibler
- ein allfälliger weiterer Programmteil, z.b. eine grafische Oberfläche muss kein spezielles Interface implementieren, eine Verbindung zur Datenbank genügt

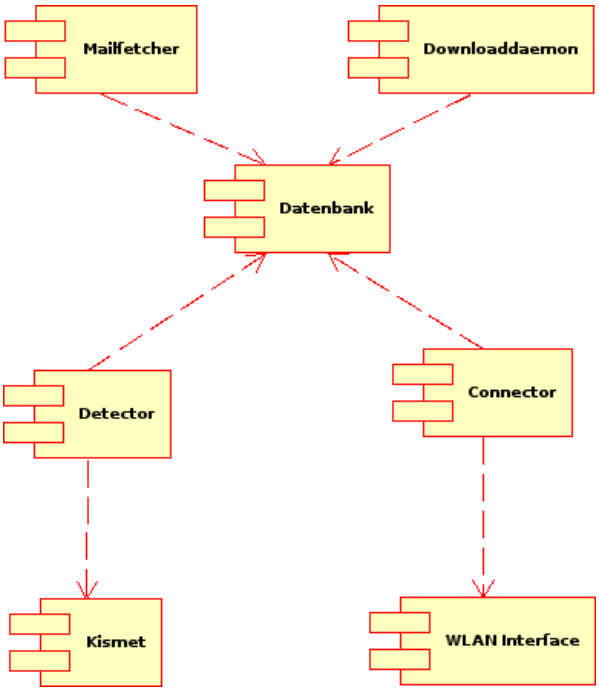


Abbildung 7.1: Interprozesskommunikation über die Datenbank

- Sehr einfaches Debugging, der Entwickler kann jederzeit mit SQL den Zustand der Applikation prüfen, ändern, speichern, wiederherstellen etc.
- Durch Views und Triggers können diverse Operationen bereits auf der Datenbank ausgeführt werden, z.b. die Erkennung möglicher Netzkandidaten. Dies bringt Performancevorteile.

Ein erster Versuch wurde mit mySQL gemacht, es stellte sich aber heraus, dass der Python-Treiber für eine hohe Anzahl SQL-Statements in kurzer Zeit nicht geeignet ist. Daher wird postgreSQL verwendet, welches mySQL in praktisch jeder Hinsicht übertrifft (Stabilität, Features, Lizenzmodell, ...). Da es sich hier primär um ein Projekt für “Geeks” handelt, können wir dem Benutzer die Installation von postgreSQL zumuten.

7.2.2 Klassen Design

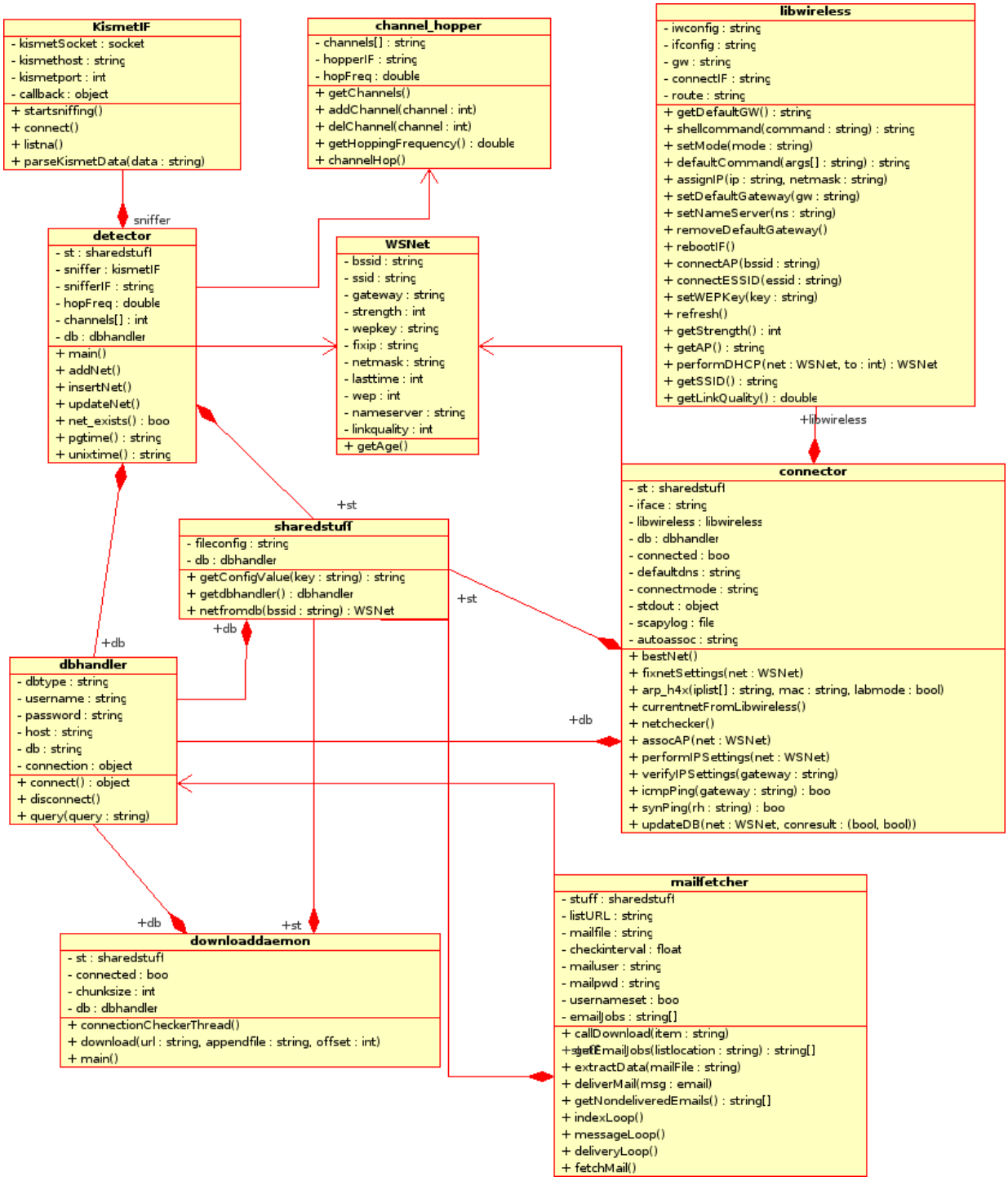


Abbildung 7.2: Klassendiagramm

Detector

Der Detector ist dafür zuständig, Momentan verfügbare Netzwerke zu erkennen. Er kommuniziert direkt über eine Socketverbindung mit Kismet, um die benötigten Informationen zu erhalten. Welche er unverändert allen anderen Komponenten zur Verfügung stellt.

Kismet-Interface

Das Netzwerkprotokoll von Kismet ist leider nur sehr spärlich dokumentiert. Grundsätzlich funktioniert es folgendermassen: Der Client (in unserem Fall der Detector) öffnet eine Socketverbindung zum Kismet server auf Port 2501. Kismet schickt dann eine Kurzübersicht über die Verfügbaren Befehle:

```
*KISMET: 0.0.0 1132307154 Kismet 20050815211952 1 2005.08.R1
*PROTOCOLS: KISMET,ERROR,ACK,PROTOCOLS,CAPABILITY,TERMINATE,TIME,ALERT,NETWORK,CLIENT,GPS,
```

Der Client meldet nun, über welche Events er informiert werden möchte und welche Informationen in dabei genau interessieren. Dies geschieht mittels folgender Syntax:

```
!<id> ENABLE <protokoll> <informationen>
```

Die id kann dabei vom Client beliebig gewählt werden. Der Detector interessiert sich für alle verfügbaren Netzwerke. Dabei benötigt er:

- BSSID
- SSID
- Kanal
- Signalstärke
- Verschlüsselung
- IP-Range
- Erkennungsart
- Zeitpunkt der letzten Sichtung

Mit Erkennungstyp ist gemeint, auf welche Art Kismet ein Netzwerk gefunden hat. Dies kann einerseits durch normale Pakete vom AP passieren, andererseits aber auch durch sog. Probe-Requests, d.h. Clients, welche ein Netzwerk suchen. Diese Probe Requests müssen wir herausfiltern, da z.b. auch die Connector Karte solche Pakete sendet und somit unseren Verbindungsalgorithmus stören würde.

Der Detector meldet sich also bei kismet mit folgendem Request an:

```
!1234 ENABLE NETWORK lasttime,bssid,wep,signal,rangeip,type,channel,ssid
```

Ab sofort sendet Kismet periodisch die gewünschten Informationen zurück. Beispiel:

```
*NETWORK: 1132315313 00:0C:F1:0A:9E:D3 0 213 0.0.0.0 2 0 Wireless
```

Diese Daten werden vom Detector in SQL Statements umgewandelt und in die Datenbank(Tabelle kismetdata) eingetragen. Bereits bestehende Netzwerke werden aktualisiert. Ein Trigger in der Datenbank überträgt diese Rohinformationen in eine weitere Tabelle (availlog), in der später nachvollzogen werden kann, welches Netzwerk wann und wie lange verfügbar war.

Channel Hopper

Die Detector Karte muss zwischen den verschiedenen Kanälen herumspringen, um möglichst viele Netzwerke zu finden. Dieses Channel Hopping kann direkt in Kismet konfiguriert werden. Bei einigen Karten wird dies von kismet aber nicht unterstützt. In diesem Fall kann versucht werden, mit dem Channel Hopper des Warsurfers zu arbeiten. Leider gibt es diverse Karten bei denen die Firmware selbst entscheidet, welche Kanäle sie anspringt, in diesem Fall bringt auch dieser Hopper keine besseren Resultate.
Die Funktionsweise des Channelhoppers ist einfach: Die Kanalsequenz und Sprungfrequenz wird in der Datenbank konfiguriert. Der Hopper benutzt das iwconfig Kommando, um den Kanal der Karte zu ändern:

```
iwpriv <karte> monitor 1 channel <kanalnummer>
```

Connector

Die Aufgabe des Connectors ist es, Verbindungen mit Accesspoints herzustellen und diese so lange wie möglich zu erhalten. Das “beste” Netz wird mittels folgendem Algorithmus ausgewählt.

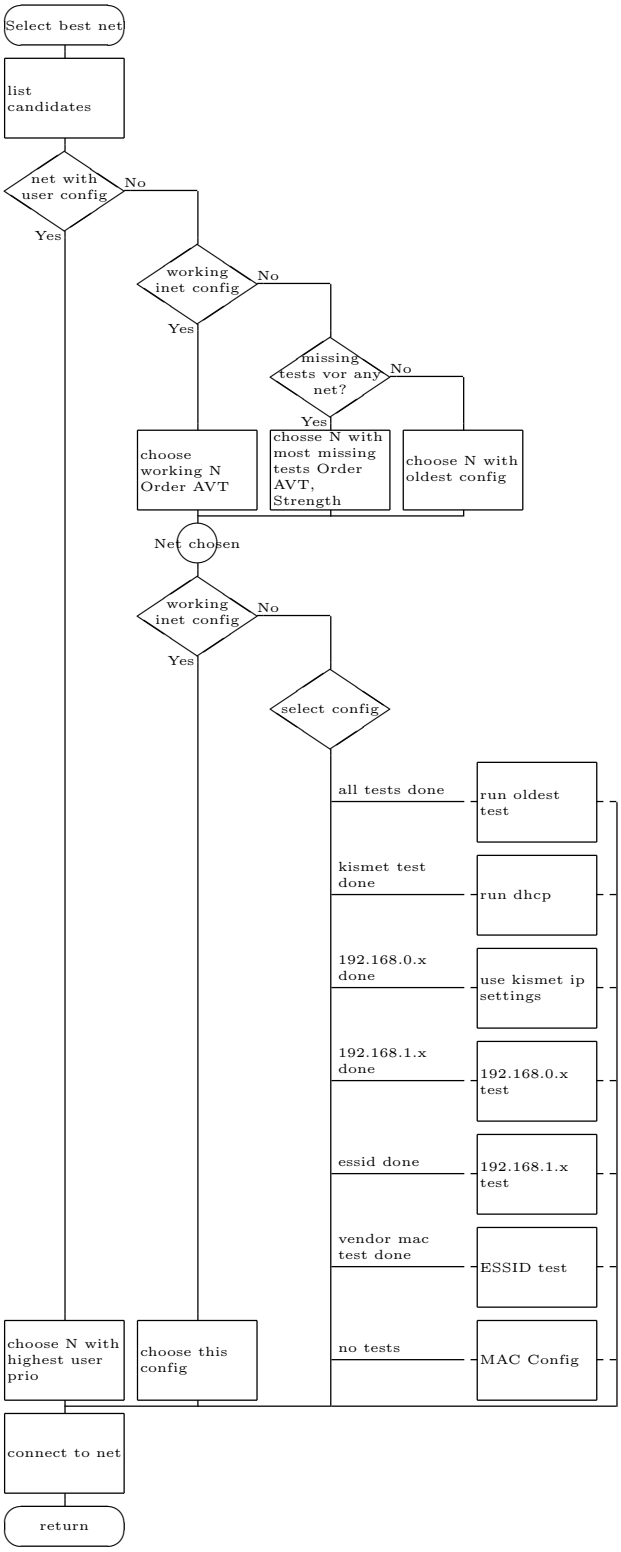

Select best net
list candidates
net with user config
No
Yes
working inet config
No
Yes
missing tests vor any net?
No
Yes
choose working N Order AVT
chosse N with most missing tests Order AVT, Strength
choose N with oldest config
Net chosen
working inet config
No
Yes
select config
all tests done
run oldest test
kismet test done
run dhcp
192.168.0.x done
use kismet ip settings
192.168.1.x done
192.168.0.x test
essid done
192.168.1.x test
vendor mac test done
ESSID test
no tests
MAC Config
choose N with highest user prio
choose this config
connect to net
return

Abbildung 7.3: Verbidungsalgorithmus

N = Netz
AVT = Average Availability Time: Durchschnittliche Zeit, die ein Netz bisher verfügbar war
Order = Treffen mehrere Netze zu, wird nach diesen Werten sortiert

Beim Start des Warsurfers wird die Verbindende Karte auf Grundwerte (Managed Modus, kein Wepkey etc) eingestellt.
Jetzt beginnt der Programmablauf. Alle Verfügbaren Netze werden aufgelistet, die nicht schon von vornherein ausgeschlossen werden können(verschlüsselt, Gebührenpflichtig etc). Enthält diese Liste ein vom Benutzer konfiguriertes Netz, werden keine weiteren Tests durchgeführt, es wird direkt verbunden.
Gibt es für ein oder mehrere Netze ein bekannte funktionierende Konfiguration, wird dieses Netz ausgewählt (bei mehreren Möglichkeiten das Netz mit der grössten AVT). Falls nicht, wird geprüft, ob Netze noch nicht alle Möglichen Tests durchlaufen haben, hier wird das Netz mit den meisten fehlenden Tests gewählt. Haben alle verfügbaren Netze alle Tests durchlaufen, wird das Netz ausgewählt, bei dem irgend ein Test am weitesten zurück liegt, in der Hoffnung, das dieser mittlerweile funktioniert.
Wir sind nun am Punkt "net choosen"d.h. wir haben ein Netz ausgewählt, auf das verbunden werden soll. Ist für das Ziel bereits eine funktionierende Konfiguration bekannt, wird diese ausgeführt. Ansonsten werden diverse Testkonfigurationen überprüft. (In der Grafik sind diese invers formuliert!)

1. Herstellersettings anhand der BSSID (Mac-Adresse) ermitteln
2. Herstellersettings anhand der ESSID (Netzname) erraten , z.b. Wireless, linksys, USR8054 etc.
3. 192.168.1.1 als Gateway annehmen, dieses Setting ist am weitesten verbreitet
4. 192.168.0.1 als Gateway annehmen, dieses Setting ist ebenfalls sehr gängig
5. Allfällig erkannte IP-Settings von kismet probieren - diese sind aber öfters falsch
6. DHCP Request
7. Sind alle Tests fehlgeschlagen, wird der Test gewählt, der am weitesten zurückliegt

Die gewählte Test wird ausgeführt. Es wird geprüft ob eine Assoziation mit dem Accesspoint funktioniert hat und ob es eine Verbindung ins Internet gibt. Die Resultate werden für den nächsten Durchlauf gespeichert. Funktioniert die Internetverbindung, wird dieses Netz so lange wie möglich gehalten (Prüfe Verbindunsqualität auf AP). Sobald die Verbindung verloren geht, beginnt die Suche von neuem.

Um die Verbindung ins Internet zu erkennen, werden zwei Arten von Pingpaketen an diverse Hosts im Internet geschickt. Einerseits normale ICMP Echo

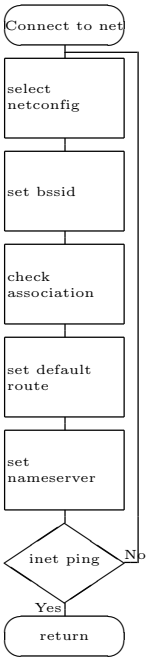


Abbildung 7.4: Verbindung mit dem Internet

Requests, andererseits sog. SYN-Pings, d.h. Verbindungsanfragen auf den Port 80 (HTTP), die mit mit SYN/ACK Paketen beantwortet werden. Diese Doppelspurigkeit soll sicher stellen, dass die Erkennung auch funktioniert, wenn ICMP von Firewalls blockiert wird.
Falls die Datenbank eine Netzwerkkonfiguration liefert, die früher einmal funktioniert hat, beim aktuellen Versuch aber fehlschlägt wird diese Konfiguration gelöscht, so erhält sie beim nächsten Mal automatisch eine höhere Priorität.

libwireless

Diese Library dient dazu die Kommunikation zwischen dem Connector und der Hardware sicherzustellen. Sie bietet Methoden an um IP-Settings auf die Wireless-Karte zu schreiben, Netzmasken zu setzen, sich mit einem Accesspoint über dessen BSSID oder ESSID zu verbinden. Sie stellt Methoden zur Verfügung um die Routing-Table an die wechselnden Gegebenheiten anzupassen, einen DHCP-Request abzusetzen oder den Nameserver festzulegen. Selbstverständlich können auch alle Werte, die verändert werden können, auch ausgelesen werden. Um die gewünschten Einstellungen vorzunehmen werden sowohl die Bibliotheken dnet und scapy, als auch die Linux-Tools ifconfig und iwconfig

- Mit ESSID verbinden
  Normalerweise wird nicht direkt über die MAC-Adresse des Accesspoint eine Verbindung hergestellt. Obwohl dies möglich ist, hat die Praxis gezeigt, dass es weniger fehleranfällig ist, wenn man stattdessen die ESSID des Accesspoint verwendet um auf diesen zu verbinden.

```
iwconfig <Karte> essid <ESSID>
```

- Mit BSSID verbinden
  Obwohl es aus Sicht der Datenhaltung, sowie des Programmierers, einfacher wäre, den Accesspoint direkt über seine MAC-Adresse anzusprechen, wurde diese Variante verworfen. Es zeigte sich in der Praxis, dass alle Wireless Interfaces, die zur Verfügung standen nicht mit einem Accesspoint verbinden, wenn direkt über die BSSID versucht wird eine Assoziation aufzubauen.

```
iwconfig <Karte> ap <BSSID>
```

- Route setzen Jedesmal, wenn mit einem neuen Accesspoint verbunden wurde, muss dieser als Standardgateway gesetzt werden. Um dies zu bewerkstelligen wird libdnet eingesetzt. Libdnet stellt ein einfaches, portables Interface zu unterschiedlichen low-level Netzwerk-Routinen. Unter anderem kann mit libdnet eben auch die Routingtable verändert werden.

```
libdnet.route().add(dnet.addr('0.0.0.0/0'), dnet.addr(<GATEWAY>))
```

- IP Adresse setzen
  Dnet bietet Mechanismen an, die IP-Adresse und die Subnetzmaske zu setzen. Setzen von IP-Adresse und Subnetzmaske geschieht in mehreren Schritten:

  1. Eine Referenz auf das Interface erstellen

     ```
     intf=dnet.intf(<Karte>)
     ```

  2. Ein Adress-Objekt erstellen

     ```
     addr=dnet.addr(<IP-Adresse>)
     ```

  3. Die aktuellen Werte der Karte in ein Dictionary-Objekt auslesen und dieses verändern

     ```
     dict=intf.get()
     dict['addr']=addr
     ```

  4. Die neuen Werte wieder auf die Karte zurückschreiben

     ```
     intf.set(dict)
     ```

- Wireless Interface rebooten
  Um sicher zu gehen, dass nicht alte Einstellungen übernommen werden, oder allenfalls Assoziationen zu nicht konfigurierten Accesspoints zu falschen Resultaten führen, wird das Wireless Interface jeweils neu gestartet, bevor eine neue Konfiguration darauf geschrieben wird. Um eine Konfiguration auf das Interface zu schreiben, werden die Linux-Tools iwconfig und ifconfig verwendet.

```
iwconfig <Karte> essid off mode managed key off
ifconfig <Karte> down
ifconfig <Karte> up
iwconfig <Karte> essid off mode managed key off
```

Um ganz sicher zu gehen, dass die Einstellungen auf der Karte wirklich gelöscht werden, wird das iwconfig-Kommando zweifach aufgerufen. Es könnte nämlich sein, dass sich die Karte, sobald das Interface wieder aufgeschaltet wird, automatisch mit einem in der Nähe befindlichen Accesspoint verbindet.

- DHCP-Request absetzen
  Wenn Keine passenden Einstellungen für ein Netz gefunden werden, wird versucht per DHCP gültige Einstellungen zu erhalten. Die meisten DHCP-Clients unter Linux (Bsp. dhclient) werden nach einem Request nicht terminiert, sondern laufen im Hintergrund weiter. Hier kommt scapy zum Einsatz. Scapy bietet die Möglichkeit DHCP-Requests zu verschicken und dabei einen Timeout festzulegen, nachdem der Versuch abgebrochen werden soll.

  ```
  ans=scapy.dhcp_request(<Karte>,to=<Timeout>)
  ```

- Werte auslesen
  Libwireless hält sich stets die Ausgabe des letzten Aufrufs von iwconfig gespeichert. Diese kann über den Aufruf der "refresh"-Methode der Klasse erneuert werden.

WSNet

WSNet ist eine Datenklasse, welche die Eigenschaften eines möglichen Netzes hat. Diese Klasse dient lediglich dazu Informationen zu halten und hat lediglich eine einzige Funktionalität; nämlich die Berechnung des Alters eines Netzes.

dbhandler

Der DBHandler ist die Schnittstelle zur Datenbank. Er baut eine Verbindung zur Datenbank auf, kann SQL-Queries übermitteln und die erfragten Daten in einem Cursor-Objekt zurückgeben.

sharedstuff

Diese Klasse bietet eine Abstraktionsstufe zum DBHandler. In dieser Klasse wird ein DBHandler Objekt erzeugt, auf welches von ausserhalb der Klasse zugegriffen werden kann. Die Klasse wurde vor allem auf die Abfrage von Konfigurationswerten, welche sich in der Datenbank befinden, optimiert. So kann mit dem entsprechenden Schlüssel direkt der entsprechende Wert abgefragt werden, ohne die SQL-Queries in einer anderen Klasse erneut absetzen zu müssen.

mailfetcher

Der Mailfetcher kommuniziert mit dem Server, welcher die Email-Nachrichten von den konfigurierten IMAP oder POP3 Servern herunterlädt und in ein Indexfile parst. Der Mailfetcher lädt nun dieses Indexfile periodisch vom Server. Anhand der im Indexfile enthaltenen Informationen entscheidet der Mailfetcher, welche Nachrichten noch nicht heruntergeladen worden sind. Er schreibt die URL derjenigen Nachrichten in die Datenbank, damit der Downloadmanager diese Filtes herunterladen kann, sobald eine Verbindung mit dem Internet besteht. Ist eine Nachricht heruntergeladen, wird sie, je nach Konfiguration, an die Datei /var/spool/mail/username oder /home/username/mbox oder sonst eine Datei angehängt, von wo aus sie mit einem Mailclient oder, bei entsprechender Konfiguration, mit dem "mail"-Kommando gelesen werden kann.

7.2.3 Datentransfer

Download Daemon

Wenn der Detector und der Connector laufen, besteht wann immer möglich eine Verbindung ins Internet. Diese wird aber bei einem Wechsel des Netzwerkes unterbrochen und führt dazu, das grosse Downloads neu gestartet werden müssen. Der Download daemon löst dieses Problem, in dem er grosse Downloads in Einzelstücke aufteilt und am Schluss wieder zu einer ganzen Datei zusammensetzt. Die Downloadaufträge erhält der Daemon(downloaddaemon.py) über die Datenbanktabelle "downloadjobs". Ist ein Job vorhanden, prüft der Daemon periodisch, ob eine Verbindung ins Internet besteht (mittels Sicht "vconnectionstate") und fordert die gewünschte Datei beim Webserver an. Wird die Internet Verbindung unterbrochen, wartet der Daemon, bis diese wieder besteht und fordert die Datei neu an. Dabei wird der [12] HTTP Range Request-Header verwendet, um den Download an der Stelle fortzusetzen, an der er beim letzten Versuch abgebrochen ist. Der Verlauf des Downloads wird in der Datenbanktabelle aktualisiert.
Dem Benutzer steht ein simples Interface(wsdownload.py) zur Verfügung, mit dem er grössere Dateien herunterladen kann.

```
python wsdownload.py <url>
```

Mail fetcher

Die Aufgabe des Warsurfers besteht ja hauptsächlich daraus, während einer Busfahrt laufend die Emails herunterzuladen. Hier sind vor allem grössere Emails mit Anhängen ein Problem, da auch hier abgebrochene Downloads wieder fortgesetzt werden müssen, was von den gängigen Mailprotokollen nicht unterstützt wird. Deshalb haben wir entschieden, die Mails über HTTP zu tunneln. Dazu ist eine Installation auf einem Webserver notwendig.

Home Agent: Auf dem Home Agent werden die Emails empfangen, komprimiert und in einem Webverzeichnis abgelegt. Der Dateiname wird dabei so gewählt, das er der MD5 Summe des Mailinhalts entspricht. Dies ermöglicht

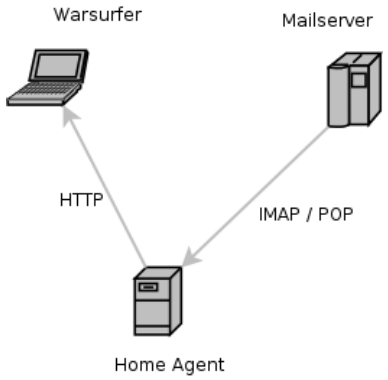


Abbildung 7.5: Mailtransfer über HTTP

dem Client zu überprüfen, ob das Mail vollständig und korrekt heruntergeladen wurde. Die Liste der Emails wird zusätzlich in einer Indexdatei abgelegt. So kann der Client einfach erfahren, welche Mails noch heruntergeladen werden müssen.
Der Aufbau der Indexdatei wurde dabei so gewählt, dass er eine einfache Erweiterung auf andere Job-Typen (z.b. Instant-Messages) zulässt. Jede Zeile im Index hat folgendes Schema:

```
<md5 summe><tab><jobtyp><tab><Grösse in bytes><URL>
```

Beispiel:

```
07eb51d640b9d9ef345c300eb0101bb8          mail     287      http://ut.darktech.org/
      warsurfer/07eb51d640b9d9ef345c300eb0101bb8.gz
d494cb6b59adfbc9fc4eec39dceacc85          mail     295      http://ut.darktech.org/
      warsurfer/d494cb6b59adfbc9fc4eec39dceacc85.gz
c18a72ab35382761abdcf707fb284075          mail     297      http://ut.darktech.org/
      warsurfer/c18a72ab35382761abdcf707fb284075.gz
```

Client Der Mailclient lädt vom Server das Indexfile herunter welches alle empfangenen Nachrichten enthält. schritt für Schritt geht der Mailclient diese List durch und prüft stets, ob die in der Liste enthaltene Datei bereits in der Datenbank eingetragen ist. Ist dies nicht der Fall, wird sie dort eingetragen.
In der Datenbank steht ebenfalls drin,. ob eine Nachricht bereits zugestellt worden ist oder nicht. Der Mailclient prüft, ob es in der Datenbank Einträge zu Nachrichten gibt, die besagen dass eine Nachricht zwar bereits vollständig heruntergeladen, aber nicht nicht zugestellt worden sind. Wird ein entsprechender Eintrag gefunden, wird überprüft, ob die MD5 Summe der erhaltenen Nachricht mit derjenigen in der Liste vom Server übereinstimmt. Wenn dies der Fall ist, wird diese lokal zugestellt.

Roaming

Das vollständige Roaming, d.h. ohne Abbruch der TCP Verbindungen stellt sich als einfacher heraus als zuerst angenommen. Ähnlich wie im auf Seite 29 beschriebenen MobileIP-Ansatz wird ein Home-Agent dazu verwendet, die Pakete zur aktuellen Adresse des Warsurfers weiterzuleiten. Anstatt über das MobileIP-Protokoll geschieht dies aber mit der Software OpenVPN[13] , welche ein solches

Roaming unterstützt. Mit OpenVPN wird eine VPN Verbindung über UDP hergestellt.

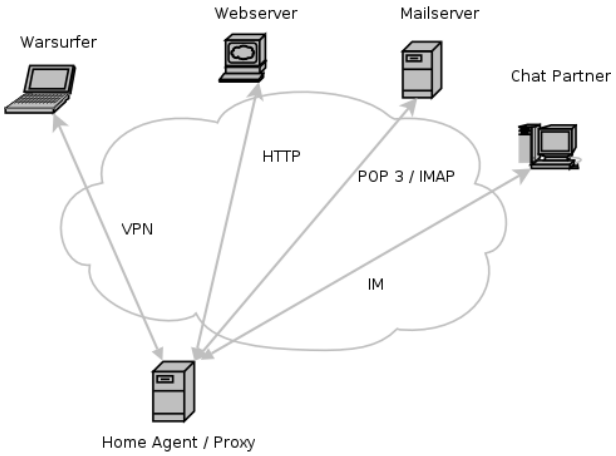


Abbildung 7.6: Roaming mit OpenVPN

Die Konfigurationen sind im Benutzerhandbuch ab Seite 74 zu finden. Es gibt nun diverse Möglichkeiten, die verschiedenen Anwendungen über diesen VPN Tunnel laufen zu lassen. Die einfachste ist es, bei erfolgreicher Verbindung die Default-Route auf den VPN Gateway zu setzen, so läuft der gesamte Verkehr über das VPN. Diese Option wird von OpenVPN Konfiguration unterstützt.
Alternativ können auch nur einzelne Protokolle über das VPN getunnelt werden. Hier bietet sich SSH-Tunneling an. Als Beispiel sei hier die Konfiguration für POP3 (Email) erklärt.
Ein Tunnel kann mit folgendem Kommando aufgebaut werden.

```
ssh -L <port>:<popserver>:<port> <benutzername>@<vpngateway>
Beispiel:\\
ssh -L 110:pop.bluewin.ch:110 warsurfer@10.1.0.1
```

Nun kann entweder der Mailreader direkt so konfiguriert werden, dass er dass er annimmt, der POP3 Server sei 127.0.0.1 , oder die IP Adresse wird mit dem Eintrag

```
127.0.0.1 pop.bluewin.ch
```

in die Datei **/etc/hosts** fest gesetzt, in diesem Fall sind im Mailreader selbst keine weiteren Konfigurationen nötig. Der Mailreader wird jetzt für den Abruf der Emails eine Verbindung zu 127.0.0.1(localhost) aufbauen. Diese wird von SSH entgegengenommen via VPN Server zu pop.bluewin.ch weitergeleitet.

7.3 Datenbank

Viele Funktionen der verschiedenen Programmteile wurden direkt in die Datenbank ausgelagert. Die hat vor allem Geschwindigkeitsvorteile und ermöglicht auch ein Eingreifen in die Funktionalität des Warsurfer, ohne das der Python-Code geändert werden muss.

7.3.1 Tabellen

kismetdata

Kismetdata ist das direkte Datenbankabbild der Informationen, die der Detector von Kismet erhält: BSSID,ESSID,Signalstärke, Verschlüsselung, und eventuell den IP-Range. Gespeichert wird zudem, wann dieses Netz zum letzten mal gesehen wurde.
Die Informationen werden vom Detector up-to-date gehalten.

```
                                Tabelle »public.kismetdata«
    Spalte     |              Typ                |                Attribute
---------------+---------------------------------+-----------------------------------------

 bssid         | character varying(17)           | not null default ''::
     character varying
 strength      | integer                         | not null default 0
 essid         | character varying(100)          |
 wepencrypted  | integer                         | not null default 0
 fixip         | character varying(15)           |
 lastseen      | timestamp without time zone     | not null
Indexe:
    »activenets_pkey« PRIMARY KEY, btree (bssid)
```

ride

Diese Tabelle enthält einen oder mehrere Rides. Sie dient dazu, Statistische Auswertungen auf eine Konkrete Fahrt zu beziehen.

```
                  Tabelle »public.ride«
   Spalte     |              Typ                | Attribute
--------------+---------------------------------+-----------
 rideid       | character varying(255)          | not null
 description  | text                            |
 transport    | character varying(200)          |
 lastrun      | timestamp without time zone     |
Indexe:
    »ride_pkey« PRIMARY KEY, btree (rideid)
```

availlog

In der Availlog Tabelle wird aufgezeichnet, welche Netze wann und wie lange verfügbar waren. Diese Information wird später verwendet, um bei mehreren verfügbaren Netzen das beste auszuwählen.
Die Werte werden durch den Trigger travaillog eingetragen.

```
                                    Tabelle »public.availlog«
  Spalte    |              Typ                |
      Attribute
----------------+---------------------------------+-----------------------------------------

 id             | integer                         | not null default nextval('
     availlog_id_seq'::regclass)
 bssid          | character varying(17)           | not null default ''::
     character varying
 essid          | character varying(100)          | not null default ''::
     character varying
 firstseen      | timestamp without time zone     |
 lastseen       | timestamp without time zone     |
 rideid         | character varying(255)          |
Indexe:
    »availlog_pkey« PRIMARY KEY, btree (id)
Fremdschlüssel-Constraints:
    »rideid_fkey« FOREIGN KEY (rideid) REFERENCES ride(rideid)
```

userconfig

Hier werden die vom Benutzer konfigurierten Netzkonfigurationen gespeichert.

Tabelle »public.userconfig«

Spalte	Typ	Attribute
usable	integer	not null default 1
wepkey	character varying(255)	
fixip	character varying(15)	
gateway	character varying(15)	
nameserver	character varying(15)	
netmask	character varying(15)	
fakemac	character varying(17)	
comment	text	
bssid	character varying(17)	
essid	character varying(100)	
priority	integer	default 100

testconfigs

Um die IP-Konfiguration eines Accesspoints herauszufinden, startet der Connector eine Reihe von Tests. Diese sind in dieser Tabelle definiert und können hier auch zentral ein- und ausgeschaltet werden.

Tabelle »public.testconfigs«

Spalte	Typ	Attribute
testid	character varying(40)	not null
priority	integer	not null default 100
description	text	
enabled	integer	not null default 1

Indexe:
    »testconfigs_pkey« PRIMARY KEY, btree (testid)

macdefaultconfig

Diese Tabelle beinhaltet eine Reihe von Standardeinstellungen für Accesspoints, basierend auf deren MAC-Adresse.

Tabelle »public.macdefaultconfig«

Spalte	Typ	Attribute
fixip	character varying(15)	
gateway	character varying(15)	
nameserver	character varying(15)	
netmask	character varying(15)	
comment	text	
priority	integer	default 100
manufacturer	character varying(100)	
model	character varying(200)	
essid	character varying(200)	
macprefix	character varying(24)	not null

essiddefaultconfig

Diese Tabelle beinhaltet eine Reihe von Standardeinstellungen für Accesspoints, basierend auf deren ESSID

Tabelle »public.essiddefaultconfig«

Spalte	Typ	Attribute
essid	character varying(100)	not null default ''::character varying

```
 usable      | integer                  | not null default 1
 fixip       | character varying(15)    |
 gateway     | character varying(15)    |
 nameserver  | character varying(15)    |
 netmask     | character varying(15)    |
 priority    | integer                  | not null default 100
 comment     | text                     |
```

connectsettings

In dieser Tabelle werden die getesteten Konfigurationen für jedes Netzwerk gespeichert. Sie ermöglicht das Lernen von Konfigurationen basierend auf bisherigen Versuchen.

```
                                                Tabelle »public.
                                                    connectsettings«
    Spalte      |              Typ                |
                                          Attribute
--
    -------------+--------------------------------+-------------------------------------------------------------------------------

 id             | integer                        | not null default nextval
     (('connectsettings_id_seq'::text)::regclass)
 bssid          | character varying(17)          | not null
 essid          | character varying(100)         |
 fixip          | character varying(15)          |
 netmask        | character varying(15)          |
 nameserver     | character varying(15)          |
 gateway        | character varying(15)          |
 testtype       | character varying(30)          | not null
 aptest         | integer                        | default -2
 inettest       | integer                        | default -2
 lastperformed  | timestamp without time zone    |
 fakemac        | character varying(17)          |
 wepkey         | character varying(255)         |
Indexe:
    »connectsettings_pkey« PRIMARY KEY, btree (id)
Fremdschlüssel-Constraints:
    »testtype_fkey« FOREIGN KEY (testtype) REFERENCES testconfigs(testid
        )
```

connectlog

Diese Tabelle zeichnet auf, auf welchen Netzen der Warsurfer effektiv eine Verbindung ins Internet hatte. Die Tabelle wird mittels der funktion fnconnected(bssid) automatisch gefüllt.

```
                                        Tabelle »public.connectlog«
  Spalte    |             Typ              |
      Attribute
--
    ----------+------------------------------+-------------------------------------------------------------------------------

 id         | integer                      | not null default nextval('
     connectlog_id_seq'::regclass)
 bssid      | character varying(17)        | not null default ''::
     character varying
 essid      | character varying(100)       | not null default ''::
     character varying
 firstseen  | timestamp without time zone  |
 lastseen   | timestamp without time zone  |
 rideid     | character varying(255)       |
```

configuration

In dieser Tabelle werden sämtliche Einstellungen als key-value pairs gespeichert.

```
          Tabelle »public.configuration«
    Spalte     |          Typ           | Attribute
---------------+------------------------+-----------
 id            | character varying(255) | not null
 configvalue   | character varying(255) |
 defaultvalue  | character varying(255) |
 description   | text                   |
Indexe:
    »configuration_pkey« PRIMARY KEY, btree (id)
```

downloadjobs

Hier können Clients(Benutzer, mail reader) ihre gewünschten Downloads platzieren. Der Downloaddaemon hält diese Downloadinformationen aktuell.

```
                Tabelle »public.downloadjobs«
      Spalte       |            Typ             | Attribute
-------------------+----------------------------+-----------
 url               | character varying(500)     | not null
 contentlength     | integer                    |
 amountdownloaded  | integer                    |
 startedat         | timestamp without time zone |
 finishedat        | timestamp without time zone |
 filelocation      | character varying(300)     |
 commited          | integer                    | default 0
 error             | integer                    |
 lastmod           | character varying(255)     |
Indexe:
    »downloadjobs_pkey« PRIMARY KEY, btree (url)
```

7.3.2 Sichten

vkismetcandidates

Diese Sicht zeit alle aktuell verfügbaren Netze , filtert aber diejenigen heraus, die vom Benutzer oder aus der essiddefaultconfig Tabelle als unbrauchbar markiert wurden. Zusätzlich gibt sie die durchschnittliche Verfügbarkeit (AVT) und die Anzahl Testkonfigurationen, die für ein Netz bereits durchgeführt wurden.

```
          Sicht »public.vkismetcandidates«
    Spalte     |          Typ           | Attribute
---------------+------------------------+-----------
 bssid         | character varying(17)  |
 strength      | integer                |
 essid         | character varying(100) |
 wepencrypted  | integer                |
 fixip         | character varying(15)  |
 age           | interval               |
 configs       | bigint                 |
 avt           | interval               |
Sichtdefinition:
 SELECT kismetdata.bssid, kismetdata.strength, kismetdata.essid,
     kismetdata.wepencrypted, kismetdata.fixip, age(now(), kismetdata.
     lastseen::timestamp with time zone) AS age, fncountconfigs(
     kismetdata.bssid) AS configs, fnavt(kismetdata.bssid) AS avt
   FROM kismetdata
  WHERE age(now(), kismetdata.lastseen::timestamp with time zone) < '
      00:00:05'::interval AND NOT (kismetdata.bssid IN ( SELECT
      userconfig.bssid
           FROM userconfig
          WHERE userconfig.usable = 0 AND userconfig.bssid IS NOT NULL))
               AND NOT (kismetdata.essid IN ( SELECT userconfig.essid
            FROM userconfig
           WHERE userconfig.usable = 0 AND userconfig.essid IS NOT NULL))
               AND NOT (kismetdata.essid IN ( SELECT essiddefaultconfig
              .essid
            FROM essiddefaultconfig
```

```
WHERE essiddefaultconfig.usable = 0)) AND NOT (kismetdata.
    bssid IN ( SELECT connectlog.bssid
   FROM connectlog
  WHERE age(now(), connectlog.lastseen::timestamp with time zone
    ) < '00:00:05'::interval)) AND NOT (kismetdata.
    wepencrypted <> 0 AND NOT (kismetdata.bssid IN ( SELECT
    userconfig.bssid
   FROM userconfig
  WHERE userconfig.usable = 1))) AND NOT (kismetdata.essid IN (
    SELECT connectlog.essid
   FROM connectlog
  WHERE age(now(), connectlog.lastseen::timestamp with time zone
    ) < '00:00:05'::interval));
```

vbestnet

Diese Sicht ruft eine Reihe von Funktionen auf und gibt eine Testkonfiguration für das "beste" momentan verfügbare Netz zurück, gemäss dem zuvor definierten Algorithmus. Der Connector nutzt diese View, um sich so mit einem Accesspoint zu verbinden.

Sicht »public.vbestnet«

Spalte	Typ	Attribute
essid	character varying(100)	
bssid	character varying(17)	
fakemac	character varying(17)	
wepkey	character varying(255)	
fixip	character varying(15)	
netmask	character varying(15)	
gateway	character varying(15)	
nameserver	character varying(15)	
netconfigid	integer	
testtype	character varying(40)	

Sichtdefinition:

```
SELECT fnchoosenet.essid, fnchoosenet.bssid, fnchoosenet.fakemac,
    fnchoosenet.wepkey, fnchoosenet.fixip, fnchoosenet.netmask,
    fnchoosenet.gateway, fnchoosenet.nameserver, fnchoosenet.
    netconfigid, fnchoosenet.testtype
  FROM fnchoosenet() fnchoosenet(essid, bssid, fakemac, wepkey, fixip,
      netmask, gateway, nameserver, netconfigid, testtype)
 WHERE fnchoosenet.bssid IS NOT NULL OR fnchoosenet.essid IS NOT NULL;
```

vconnectionstate

Mittels dieser Sicht können sich Komponenten darüber informieren, ob das System momentan mit dem Internet verbunden ist oder nicht.

Sicht »public.vconnectionstate«

Spalte	Typ	Attribute
id	integer	
bssid	character varying(17)	
essid	character varying(100)	
firstseen	timestamp without time zone	
lastseen	timestamp without time zone	
rideid	character varying(255)	

Sichtdefinition:

```
SELECT connectlog.id, connectlog.bssid, connectlog.essid, connectlog.
    firstseen, connectlog.lastseen, connectlog.rideid
  FROM connectlog
 WHERE age(now(), connectlog.lastseen::timestamp with time zone) < '
     00:00:05'::interval;
```

7.3.3 Funktionen

fnavaillog

Triggerfunktion, welche Änderungen an kismetdata in die availlogtabelle überträgt. Ist ein Netz zum ersten mal oder nach einiger Zeit wieder verfügbar, wird eine neue Zeile eingetragen. Bei bereits aktiven Netzen wird die lastseen spalte aktualisiert. Die Ride-ID wird ebenfalls eingetragen, um später eine Aufschlüsselung nach Fahrzeug zu ermöglichen.

fnconnected

Diese Funktion wird vom Connector aufgerufen um der Datenbank mitzuteilen, dass eine Internetverbindung mit dem übergebenen Netz möglich war. Sie erstellt dann einen neuen Eintrag in connectlog oder aktualisiert den bestehenden, falls es sich immernoch um das selbe Netz wie beim letzten Aufruf innerhalb der letzten 5 Sekunden handelt.
Parameter: bssid

fngetessid

Diese Hilfsfunktion wandelt eine bssid mittels der Informationen aus kismetdata in eine ESSID um.
Parameter: bssid

fnavt

Berechnet die Durchschnittliche Verfügbarkeit eines Netzes mittels der Informationen aus availlog.
Parameter: bssid

fncountconfigs

Zählt die Anzahl durchgeführter (aktiver) Tests für ein Netz. So kann ermittelt werden, ob für ein verfügbares Netz noch nicht alle Konfigurationen getestet wurden.

fntestmac

Generiert eine MAC-Testkonfiguration. Diese Funktion versucht mittels der Informationen aus macdefaultconfig eine Standardkonfiguration für die übergebene BSSID zu finden. Parameter: bssid

fntestessid

Generiert eine ESSID Testkonfiguration. Diese Funktion versucht mittels der Informationen aus essiddefaultconfig eine Standardkonfiguration für die übergebene BSSID zu finden.
Parameter: bssid

fntest19216811

Generiert eine Konfiguration für ein 192.168.1.x. Subnet.
Parameter: bssid

fntest19216801

Generiert eine Konfiguration für ein 192.168.0.x. Subnet.
Parameter: bssid

fntestkismet

Generiert eine Konfiguration aus dem IP-Range, der von kismet erkannt wurde.
Parameter: bssid

fntestdhcp

Generiert eine leere DHCP Konfiguration.
Parameter: bssid

fngettest

Sucht den nächsten fehlenden Test für das übergebene Netzwerk und gibt die gefundene Konfiguration zurück. Fehlt kein Test wird der älteste Test wiederholt.
Parameter: bssid

fnchoosenet

Sucht sich aus allen verfügbaren Netzen das tendenziell "beste" (gemäss obigen Algorithmus) aus und gibt die nächste Testkonfiguration für dieses Netz zurück.
Parameter: keine

Teil III

Systemanwendung

Kapitel 8

Warsurfer HOW-TO

In diesem Kapitel lernen Sie, wie Sie Warsurfer auf ihrem System installieren und anwenden können.

8.1 Über die Warsurfer Installation

Willkommen bei Warsurfer. Vorbei sind die Zeiten, in denen Tram- oder Busfahrten eine schier unerträgliche Absenz von Internet und Email bedeuteten. Mit dem Warsurfer gehören diese Sorgen der Vergangenheit an.
Die Installation ist zwar nicht ganz einfach. Dieses How-To gibt aber klare Anweisungen und führt den Leser Schritt für Schritt durch die Installation und Konfiguration der für den Warsurfer benötigten Komponenten.
Als erstes werden die Abhängigkeiten, die der Warsurfer hat, behandelt. Es wird gezeigt, welche Programme und Libraries zusätzlich installiert werden müssen und welche Schritte dazu nötig sind. In einem weiteren Schritt wird dann auf die eigentliche Installation des Warsurfers eingegangen. Zum Schluss werden die nötigen Konfigurationen geschrieben.
Nachdem der Warsurfer komplett installiert und richtig konfiguriert worden ist, wird gezeigt, wie er zu handhaben ist. Zum Schluss wird noch auf verschiedene Tuning-Möglichkeiten hingewiesen, die das mobile Surf-Erlebnis steigern können.

8.2 Abhängigkeiten

User, die den Warsurfer auf einem Debian-System installieren möchten, brauchen nicht alle Pakete einzeln zu installieren. Warsurfer kann mittels apt-get installiert werden. Alle Abhängigkeiten werden so automatisch aufgelöst und mit installiert.
Um den Warsurfer mittels apt-get installieren zu können, muss die Datei sources.list um folgende Zeile erweitert werden:

```
deb http://www.wgwh.ch/debian/ binary/
```

Benutzer auf Red Hat/Fedora Systemen halten am besten nach entsprechenden rpm-Paketen Ausschau. Meist wird auf den Pages, auf denen der Source-

code zum Download angeboten wird, auch ein entsprechendes RPM-Paket zur Verfügung gestellt.

8.2.1 Python

Die gesamte Applikation ist in Python geschrieben worden. Um den Warsurfer ausführen zu können muss deshalb zwingend Python 2.3 installiert sein. Mit anderen Python-Versionen funktionieren nicht mehr alle Libraries, die vom Warsurfer verwendet werden.

Installation von Python

Python ist weit verbreitet, weshalb beinahe jede Distribution die für Sie gängigen Paketformate bereitstellt. Achten Sie bei der Installation darauf, dass Sie die Version 2.3 installieren. Installation des Sourcecodes Um Python von Hand zu kompilieren und zu installieren muss als erstes der Sourcecode heruntergeladen werden. Der Quellcode wird von den Python Entwicklern unter www.python.org zur Verfügung gestellt. Sobald sich der Sourcecode auf ihrer Festplatte befindet, können Sie mit der eigentlichen Installation beginnen.
Wechseln Sie in das Verzeichnis in welchem sich das komprimierte Archiv befindet.

```
cd /pfad/zu/ihrem/Archiv/
```

Nun entpacken Sie das Archiv

```
tar -xzf Python-2.3.<Version>.tgz
```

oder

```
tar -xjf Python-2.3.<Version>.tar.bz2
```

Beim Entpacken des Archivs wurde ein Verzeichnis analog zum Dateinamen "Python-2.3.¡Version¿" angelegt. Wechseln Sie in dieses Verzeichnis und nehmen die Konfiguration vor. Für fortgeschrittene Optionen lesen Sie bitte die README-Datei.

```
./configure
```

Sobald die Konfiguration abgeschlossen ist, kompilieren Sie den Code um eine ausführbare Datei zu erhalten und diese mit Root Rechten zu installieren.

```
make
```

und als root

```
make install
```

8.2.2 dnet

Dnet ist eine Library, die eine Lowlevel-Schnittstelle zu unterschiedlichen Netzwerk-Interfaces zur Verfügung stellt. Für die Installation ist zwingend gcc Version 3.3 nötig. Die Kompilierung von dnet mit gcc 4.0 schlägt fehl. Als erstes laden Sie sich den Quellcode von "http://prdownloads.sourceforge.net/libdnet/libdnet-1.10.tar.gz?download" herunter.

```
wget http://prdownloads.sourceforge.net/libdnet/libdnet-1.10.tar.gz?download
```

Entpacken Sie die Archivdatei

```
tar -xvzf libdnet-1-10.tar.gz
```

Wechseln Sie in das neu angelegte Verzeichnis

```
cd libdnet-1.10
```

Führen Sie die Installation mit den folgenden Schritten durch

```
./configure
make
make install
cd python
python setup.py install
```

Der letzte Schritt ist wichtig, um die Python-Erweiterungen für dnet zu installieren.

8.2.3 scapy

Scapy ist eine Library, mit welcher es möglich ist, eine Vielzahl von Paketen zu erzeugen und diese zu verschicken. Die Installation gestaltet sich relativ einfach. Laden Sie sich die neueste Version von Scapy herunter und speichern Sie die Datei im Verzeichnis /usr/bin. Die Datei wird nicht nach /usr/lib/python2.3/site-packages kopiert, weil das Installationsskript des Autors diese Datei nach /usr/bin kopiert, obwohl dies der Regel entspricht.

```
wget http://www.secdev.org/projects/scapy/files/scapy.py
```

Und als root:

```
mv scapy.py /usr/bin
chmod 755 /usr/bin/scapy.py
```

Scapy bietet die Möglichkeit direkt über die Konsole Pakete zu erstellen und zu verschicken. Diese Möglichkeit ist vor allem zur Fehlersuche sehr komfortabel. Möchten Sie davon gebrauch machen, erstellen Sie einen Link im Verzeichnis /usr/bin.

```
ln -s /usr/bin/scapy.py /usr/bin/scapy
chmod 755 /usr/bin/scapy
```

8.2.4 Postgresql

PostgreSQL ist, wie MySQL auch, eine Datenbank. Im Unterschied zu MySQL besticht PostgreSQL aber durch grössere Stabilität, mehr Möglichkeiten, was SQL betrifft und vor allem ein weniger restriktives Lizenzmodell. Diese Datenbank bildet das Herzstück des Warsurfers. Der Sourcecode wird mit folgendem Befehl heruntergeladen:
(Achten Sie darauf, dass Sie mindestens Version 8.1 von Postgres installieren)

```
wget ftp://ftp2.ch.postgresql.org/pub/postgresql/source/v8.1.0/postgresql-8.1.0.tar.bz2
```

Sobald Sie den Sourcecode heruntergeladen haben, entpacken Sie diesen und wechseln ins neu angelegte Verzeichnis

```
tar -xjf postgresql-8.1.0.tar.bz2
cd postgresql-8.1.0
```

Nehmen Sie die Konfiguration und die Installation mit folgenden Kommandos vor:

```
./configure
make
```

Installieren Sie PostgreSQL schliesslich als Root mit folgendem Befehl:

```
make install
```

8.2.5 psycopg

Psycopg ist ein PostgreSQL Datenbank Adapter, der mit dem Fokus auf Stabilität und Performance geschrieben wurde. Die Installation hat verschiedene Abhängigkeiten. Benutzer, welche nicht Debian verwenden, sei ein Blick auf `http://initd.org/tracker/psycopg/wiki/PsycopgOne` empfohlen.

```
wget http://initd.org/pub/software/psycopg/psycopg-1.1.21.tar.gz
tar -xzf psycopg-1.1.21.tar.gz
cd psycopg-1.1.21/
./configure \
--with-postgres-libraries=‘pg_config --libdir‘ \
--with-postgres-includes=‘pg_config --includedir‘
make
```

Und schliesslich als Root:

```
make install
```

8.2.6 Kismet

Kismet ist der Sniffer für Wireless-Netzwerke schlechthin.Höchstwahrscheinlich befindet er sich bereits auf ihrem System und Sie können diesen Schritt getrost überspringen und mit der Installation des Warsurfers fortfahren. Alle die nicht in der glücklichen Lage sind, Kismet bereits installiert haben, holen dies in den nächsten Schritten nach. Als erstes wird der Sourcecode heruntergeladen und entpackt.

```
wget http://www.kismetwireless.net/code/kismet-2005-08-R1.tar.gz
tar -xzf kismet-2005-08-R1.tar.gz
```

Als nächstes nehmen Sie die Konfiguration und die Kompilation vor.

```
./configure
make
```

Schliesslich installieren Sie Kismet mit folgendem Befehl Root Rechten

```
make install
```

So! Nun gehören auch Sie zu den glücklichen Kismet Benutzern. Schliesslich haben Sie nun alles installiert, was der Warsurfer zur Funktion benötigt. Fahren Sie nun mit der Installation des Warsurfers fort.

8.3 Installation

Die Installation des Warsurfers ist einfacher als man denkt. Dies trifft vor allem für Debian-User zu. Diese können einfach auf der Konsole

```
apt-get install warsrufer
```

ausführen. Der Warsurfer wird dann unter /usr/share/warsurfer installiert. Benutzer, welche von dieser Möglichkeit nicht Gebrauch machen können, haben es nicht viel schwerer. Einfach den Sourcecode bei der Bezugsquelle ihres Vertrauens erstehen, auf die Festplatte speichern, entpacken und in das neu angelegte Verzeichnis wechseln.

```
wget http://warsurfer.wgwh.ch/download/warsurfer-version.tar.gz
tar -xvzf warsurfer-version.tar.gz
cd warsurfer-version
```

8.3.1 Datenbank(automatisch)

Im Installationsverzeichnis befindet sich ein Shell Skript, welches die Datenbank aufsetzt. (Dieses muss auch unter Debian ausgeführt werden: /usr/bin/wasurfer_installer.sh)

```
sh warsurfer_installer.sh
```

Es erscheint eine Nachricht wie in Abb 8.1.

Drücken Sie OK. Sie werden als nächstes nach dem Namen gefragt, der für die neue Datenbank verwendet werden soll. Der Standardwert ist sicher eine gute Wahl, Sie können den Namen aber auch ändern, wenn Sie z.B. mehrere unabhängige Warsurfer Datenbanken benutzen möchten. (Abb. 8.2)
Jetzt werden Sie nach einem Datenbankbenutzer gefragt, der der Eigentümer der neuen Datenbank werden soll. Auch hier ist der Standardwert eine gute Wahl. (Abb. 8.3)
Für den neuen Datenbankbenutzer muss jetzt ein Passwort festgelegt werden. Achtung, verwenden Sie hier kein Passwort, welches Sie auch an anderen Orten

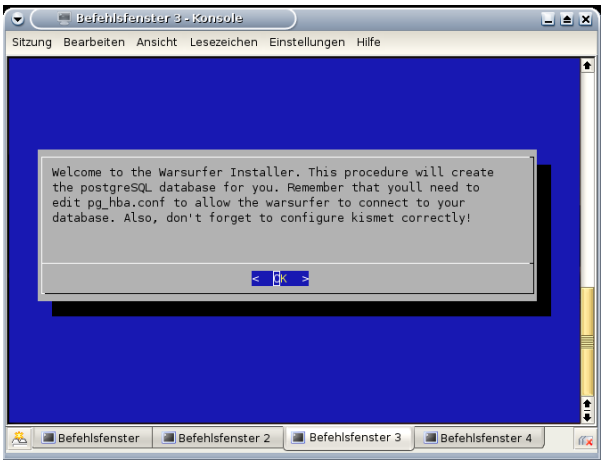


Abbildung 8.1: Installer: Einführung

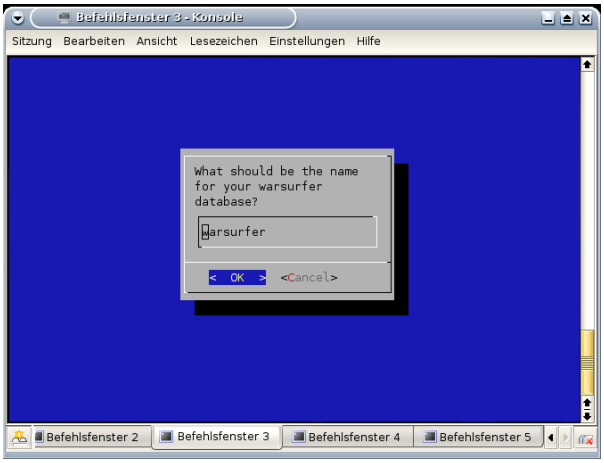


Abbildung 8.2: Installer: Datenbankname

benutzen, denn dieses Passwort kann von anderen Benutzern ihres Systems unter Umständen gelesen werden! (Abb. 8.4)

Jetzt wird die Datenbank automatisch erstellt, allfällige Fehlermeldungen können an dieser Stelle ignoriert werden. Nun geht es an die Grundkonfiguration des Warsurfers. Sie müssen als erstes den Interfacenamen der Wirelesskarte angeben, welche zum Erkennen der Drahtlosnetzwerke verwendet werden soll. (Abb. 8.5)
Danach folgt die Konfiguration der Karte, welche die Verbindung zu den gefundenen Netzwerken aufbauen soll. Dies darf nicht dieselbe Karte sein! (Abb. 8.6

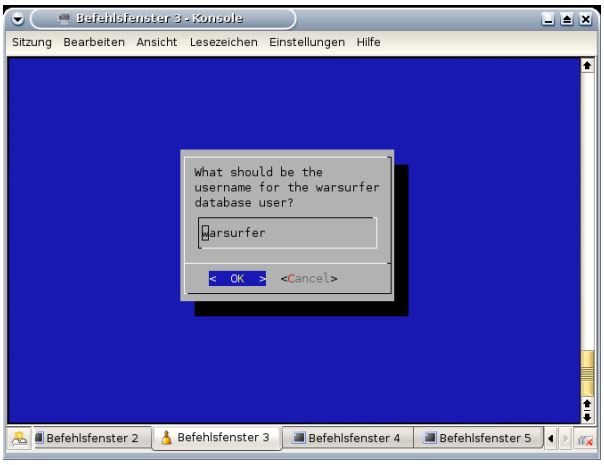


Abbildung 8.3: Installer: Datenbankbenutzer

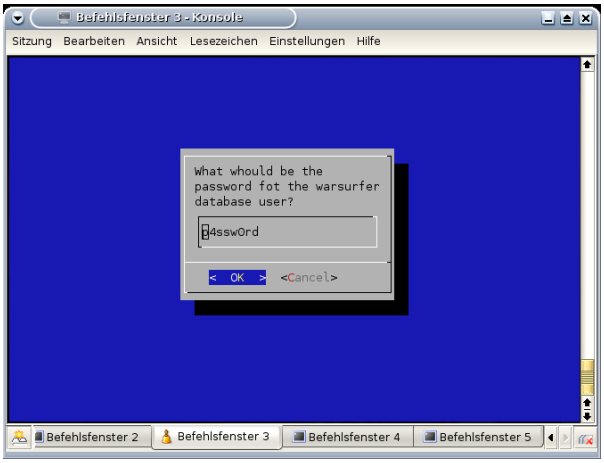


Abbildung 8.4: Installer: Datenbankpasswort

8.3.2 Datenbank(Manuell)

Sollte die automatische Prozedur fehlschlagen, oder wenn Sie dem Skript kein Vertrauen schenken, kann die Datenbank auch manuell erstellt werden:

```
su postgres
psql
create database warsurfer;
```

Danach verbinden Sie sich mit der Datenbank und erzeugen die prozedurale Sprache plpgsql.

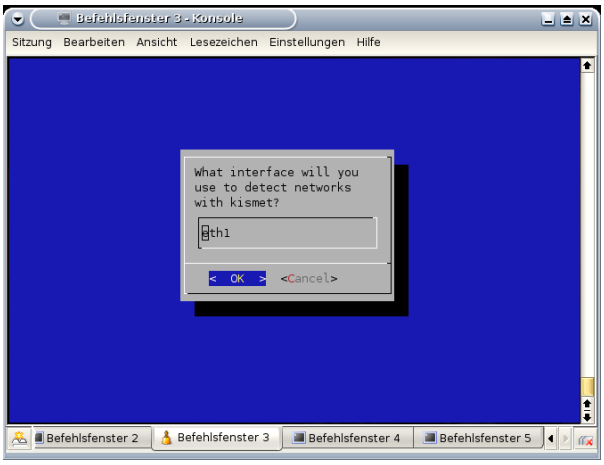


Abbildung 8.5: Installer: Detectorinterface

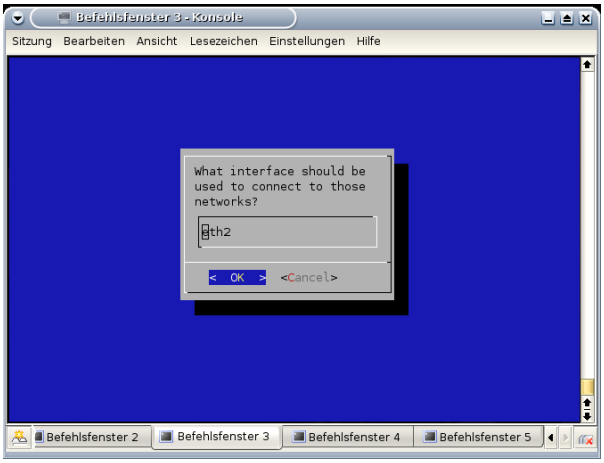


Abbildung 8.6: Installer: Connectorinterface

```
connect warsurfer;
create language plpgsql;
```

Nun legen Sie den Benutzer “warsurfer” an und geben ihm ein Passwort, sowie die benötigten Berechtigungen. Sobald dies erledigt ist, können Sie die Verbindung zur Datenbank trennen.

```
create user warsurfer;
alter user warsurfer with password 'p4ssw0rd';
alter database warsurfer owner to warsurfer;
\q
```

Damit der Warsurfer sich mit der Datenbank verbinden kann, muss er dazu die Berechtigung erhalten. Die Berechtigungen der Datenbank werden über

das File “pg_hba.conf” geregelt. Öffnen Sie die Datei mit ihrem favorisierten Texteditor.

```
vi /etc/postgresql/8.1/main/pg_hba.conf
```

Fügen Sie nun folgende Zeile in der Local-Section ein.

```
local   warsurfer   warsurfer   password
```

Um die Tabellen, Views und Funktionen der Tabelle für den Warsurfer zu erzeugen wechseln Sie in das Verzeichnis, in welchem die sql-Files sind. (Debian: /usr/share/doc/warsurfer/sql)

```
cd /path/to/warsurfer/sql
```

Verbinden Sie sich erneut mit der Datenbank. Diesmal verwenden wir den neu angelegten Benutzer “warsurfer” und führen Sie die SQL-Skripts aus.

```
psql -U warsurfer warsurfer
\i base.sql
\i connector.sql
\i detector.sql
```

8.3.3 Startskript

(Dieser Schritt ist unter Debian nicht nötig) Das warsurfer Startskript sucht seine Bibliotheken und Programmdateien zuerst in /usr/share/warsurfer , danach im aktuellen Verzeichnis. Wenn Sie also einen anderen Installationspfad gewählt haben, sollten Sie folgendes Wrapperskript anlegen:

```
#!/bin/bash
DIR="<WARSURFER_VERZEICHNIS>"
cd $DIR
./warsurfer
```

Erstellen Sie einen Symlink im Verzeichnis /usr/bin, welcher auf die neu erzeugte Datei zeigt oder verschieben Sie das neue Skript direkt nach /usr/bin.

8.4 Konfiguration

Nachdem nun alle Dateien an den richtigen Ort verschoben worden sind, ist es an der Zeit alle Anwendungen an ihre Bedürfnisse und Gegebenheiten anzupassen. Als erstes werden wir die Konfiguration des Herzstückes, der Datenbank, vornehmen, danach werden wir Kismet so anpassen, dass bei der Verwendung mit dem Warsurfer keine Komplikationen auftreten. Zum Schluss werden wir den Warsurfer konfigurieren. Dies geschieht in zwei Schritten. Die Grundlegensten Einstellungen werden in einem File Namens warsurfer.conf eingestellt, während der Grossteil der Einstellungen über die Datenbank vorgenommen wird.

8.4.1 Kismet

Die Konfiguration von Kismet geschieht über die Datei "/etc/kismet/kismet.conf". Die meisten Einstellungen stimmen bereits und können einfach übernommen werden. Wichtig ist, dass Sie Kismet verraten mit welchem Interface Sie gerne sniffen würden und dass Kismet von demjenigen Interface mit welchem jeweils eine Verbindung zu den verschiedenen Netzen aufgebaut wird, alle Pakete ignorieren soll.
Als erstes teilen wir Kismet mit, mit welcher Karte wir sniffen wollen.Öffnen Sie die Konfiguration mit ihrem Lieblings Texteditor und suchen Sie nach der Stelle wo Sie die sourcen definieren können.

```
# Sources are defined as:
# source=sourcetype,interface,name[,initialchannel]
source=<Treibername>,<Interface>,<Name>
```

Der Name kann frei gewählt werden. Er ist vor allem dann wichtig, wenn Sie mehrere Interfaces haben, die Sie mit Kismet verwenden wollen. In diesem Falle können Sie mehrere sourcen definieren, die sich jeweils durch den Namen unterscheiden lassen. Wenn Sie mehrere Sourcen definiert haben, müssen Sie Kismet noch mitteilen welche dieser Sourcen Sie jetzt auch wirklich aktiviert haben wollen. Dies geschieht über den Konfigurationseintrag "enablesource". Dies ist nicht nötig, wenn Sie nur eine Source definiert haben, oder mit allen definierten Sources sniffen wollen.

```
# Comma-separated list of sources to enable.  This is only needed if you defined
# multiple sources and only want to enable some of them.  By default, all defined
# sources are enabled.
# For example:
# enablesources=ciscosource
enablesources=<Name>
```

Während dem Betrieb des Warsurfers wird diejenige Karte, welche sich mit den Accesspoints verbindet (Connectorinterface), Pakete aussenden, die Kismet entdeckt. Diese Pakete können dazu führen dass der Warsurfer nicht korrekt funktioniert. Deshalb teilen wir Kismet mit, dass er Pakete, welche vom Connectorinterface her kommen, ignorieren soll. Dies geschieht über den Konfigurationseintrag "filter_tracker". Suchen Sie in der Konfiguration nach der Sektion in welcher der filter_tracker erklärt wird und fügen Sie unterhalb folgende Zeile ein.

```
filter_tracker=ANY(!<MACADRESSE_CONNECTORINTERFACE>)
```

8.4.2 Warurfer

Der Grossteil aller Einstellungen, die im Warsurfer möglich sind, werden in der Datenbank in der Tabelle "configurations" vorgenommen. Informationen dir nötig sind um sich aber mit der Datenbank zu verbinden stehen in der Datei "warsurfer.conf".
Als erstes behandeln wir die Werte, welche in der warsurfer.conf-Datei stehen und werden uns danach mit den Konfirgurationsmöglichkeiten, die über die Tabelle configurations möglich sind, zuwenden.

warsurfer.conf

Die Datei warsurfer.conf befindet sich im Verzeichnis "/etc". Grundsätzlich sollte die Datei vom Installationsskript warsurfer_installer.sh bereits korrekt erstellt worden sein. Falls nicht, öffnen Sie die Datei mit dem Texteditor ihrer Wahl. Achten Sie dabei darauf, dass Sie ausreichende Berechtigungen haben um in dieses Verzeichnis zu schreiben.

```
vi /etc/warsurfer.conf
```

Die Struktur der Datei ist relativ simpel gehalten. Sie setzt sich zusammen aus dem Namen des Konfigurationseintrages, gefolgt von einem Abstand und dem Konfigurationswert.

```
<Name> <Wert>
```

In dieser Datei haben Sie die Möglichkeit folgende Werte zu konfigurieren:

- host
  Hier wird eingestellt auf welchem Host sich die Datenbank befindet. Der Standardwert ist "localhost" und es besteht wohl wenig Anlass dazu, dies zu ändern.

  ```
  host localhost
  ```

- user
  Dieser Wert bezeichnet der Benutzernamen des Benutzers, mit welchem der Warsurfer sich mit der Datenbank verbindet. Sofern die Datenbank durch das Installationsskript oder durch obige Anleitung erzeugt worden ist, brauchen Sie den Standardwert auch nicht zu verändern.

  ```
  user warsurfer
  ```

- dbname
  Hier können Sie angeben, wie die Datenbank, die die Tabellen für den Warsurfer enthält, heisst. Auch hier gilt: Sollte die Datenbank mit den mitgelieferten Skripts erzeugt worden sein, brauchen Sie hier nichts zu verändern.

  ```
  dbname warsurfer
  ```

- password
  Geben Sie hier das Passwort ein, welches für den Datenbankbenutzer angelegt worden ist.

  ```
  password p4ssw0rd
  ```

Konfigurationstabelle

Nun sind alle Einstellungen in der warsurfer.conf so eingestellt, dass der Warsurfer Verbindung mit der Datenbank herstellen kann. Alle weiteren Konfigurationseinstellungen werden aus der Datenbank gelesen. Konkret handelt es sich um die Tabelle "configuration". Verbinden Sie sich mit der Datenbank und sehen Sie sich die Details dieser Tabelle an.

```
psql -U warsurfer warsurfer
\d configuration
```

Sie sehen nun die Spalten, die in dieser Tabelle enthalten sind. Nachdem Sie den Befehl ausgeführt haben, wird ihnen etwa folgendes angezeigt:

```
    Spalte    |          Typ           | Attribute
--------------+------------------------+-----------
 id           | character varying(255) | not null
 configvalue  | character varying(255) |
 defaultvalue | character varying(255) |
 description  | text                   |
```

Die Namen in der Tabelle sind relativ selbsterklärend. ID bezeichnet den Namen der Konfigurationsmöglichkeit, configvalue den aktuell konfigurierten Wert, defaultvalue den Standardwert und description eine kurze Beschreibung der Einstellung.
Nachfolgend sind alle möglichen Konfigurationseinstellungen beschrieben.Um eine Übersicht über die aktuellen Einstellungen zu erhalten, benutzen Sie die SQL-Syntax.

```
SELECT * FROM configuration;
```

Mit der selben Syntax können Sie auch Werte verändern:

```
UPDATE configuration SET configvalue='<NEUER_WERT>' WHERE id='<ID>';
```

Sehen wir uns die Möglichkeiten der Konfiguration im Detail an:

- dhcptimeout
  Ist ein Netz sichtbar, wird mit unterschiedlichen Methoden versucht, darauf zuzugreifen. Eine Methode davon ist, einen DHCP-Request abzusetzen. Dieser Wert bezeichnet, wie lange (in Sekunden) auf eine Antwort von einem DHCP-Server gewartet werden soll.

- pinginterval
  Ist einmal eine Verbindung mit dem Internet hergestellt, wird ständig überprüft, ob diese Verbindung noch aktiv ist. Dazu werden periodisch ICMP-Pings verschickt. Über den "pinginterval" kann eingestellt werden, wie gross der zeitliche Abstand (in Sekunden) zwischen zwei Ping-Sequenzen sein soll.

- searchinterval
  Es kommt vor, dass zu einem gewissen Zeitpunkt kein geeignetes Netz verfügbar ist. Der Connector wartet dann, bis wieder ein Netz sichtbar auftaucht, mit welchem versucht werden kann eine Verbindung aufzubauen. Dieser Wert stellt ein, in welchem Intervall (in Sekunden) der Connector überprüfen soll, ob schon ein Netz verfügbar ist.

- connectmode
  Von Linux werden zwei unterschiedliche Varianten unterstützt um sich mit einem Accesspoint zu verbinden. Eine Verbindung kann entweder über den Namen des Drahtlosnetzwerkes hergestellt werden, oder über direkt über die MAC-Adresse des Accesspoints. Die beiden Werte die hier eingestellt

werden können, sind also “essid” oder “bssid” wobei essid den Namen des Netzwerkes bezeichnet und bssid die MAC-Adresse des Accesspoints. Den Entwicklern ist kein Treiber bekannt, mit welchem es funktionieren würde, sich über die bssid zu verbinden. Es ist also ratsam hier den Standardwert stehen zu lassen.

- defaultdns
  Hier wird der DNS-Server konfiguriert. Standardmässig ist hier der DNS-Server von Bluewin eingestellt, weil er aus der Schweiz von jedem Netzbetreiber her erreichbar ist. Achten Sie bei der Konfiguration zwingend darauf, dass Sie einen für alle sichtbaren DNS-Server verwenden. Die Nameserver von Cablecom (62.2.17.61 und 62.2.17.124) eignen sich beispielsweise nicht, da sie aus dem Bluewin-Netz nicht erreichbar sind. Es lässt sich während einem Ride nicht vorhersagen über welchen Anbieter man ins Internet gelangt.

- mailfile
  Ein Feature des Warsurfers ist es, Email-Nachrichten herunter zu laden. Mit diesem Wert können Sie einstellen in welche Date Sie die neuen Nachrichten gerne geschrieben hätten. Linux unterstützt zwei unterschiedliche Ansätze. Kommt ein neues Email für einen Benutzer an, wird es nach /var/mail/username geschrieben. Sobald der Benutzer die Nachricht gelesen hat, wird sie in die Datei /home/username/mbox verschoben. Je nach dem, wie Sie ihre Nachrichten lesen wollen oder wie Sie ihren Mailclient eingerichtet haben, werden hier andere Einstellungen für Sie optimal sein.

- mailindexfile
  Während einem Ride verbindet sich der Warsurfer mit dem Home-Agent, um die Emails herunterzuladen. Alle Nachrichten sind dort in einer Indexdatei aufgelistet. Der Warsurfer lädt diese periodisch herunter und entscheidet, welche Emailnachrichten transferiert werden müssen. Dieser Wert beschreibt die URL der besagten Datei.

- mailindexrefreshinterval
  Dieser Wert (in Sekunden) beschreibt in welchem Intervall der Warsurfer das Indexfile herunterladen soll.

- mailhttpuser
  Dieser optionale Parameter legt fest, mit welchem Benutzernamen sich der Mailfetcher anmelden soll, um die Nachrichten herunterzuladen. Wird nur benötigt, wenn die Maildaten auf dem Server mittels .htaccess geschützt wurden.

- mailhttppassword
  Dieser optionale Parameter legt fest, mit welchem Passwort sich der Mailfetcher anmelden soll, um die Nachrichten herunterzuladen. Wird nur benötigt, wenn die Maildaten auf dem Server mittels .htaccess geschützt wurden.

- hopchannels
  Bei gewissen Treibern kann Kismet das Channelhopping nicht aktivieren.

Der Warsurfer bietet die Möglichkeit, die Detektorenkarte selber auf verschiedene Frequenzen einzustellen. Geben Sie hier an, auf welchen Kanälen die Detektorenkarte lauschen soll.

- hopspeed
  Dieser Wert (in Sekunden) besagt wie schnell zwischen den einzelnen Kanälen, die unter "hopchannels" eingestellt worden sind, gewechselt werden soll. Um das Channelhopping des Warsurfers zu deaktivieren geben Sie hier den Wert 0 ein. Diese Einstellung bezieht sich nur auf das Warsurfer-eigene Channelhopping und verändert das Verhalten von Kismet nicht. Achten Sie also darauf, dass Sie das Channelhopping des Warsurfers nur dann aktivieren, wenn es in Kismet deaktiviert oder nicht möglich ist!

- ignoreassocfail
  Am Anfang einer erfolgreichen Verbindung ins Internet steht die Assoziation mit einem Accesspoint. Standardmässig wird erst dann versucht IP-Einstellungen vorzunehmen und eine Verbindung mit dem Internet aufzubauen. Es kann eingestellt werden ob eine Erfolgreiche Assoziation mit dem Accesspoint abgewartet werden soll oder nicht. Belassen Sie die Einstellungen so, dass eine Assoziation abgewartet wird. Dieses Verfahren ist erfolgversprechender. Gültige Werte sind 0 für "nicht ignorieren" und 1 für "ignorieren".

- assoctimeout
  Mit diesem Wert (in Sekunden) wird angegeben, wie lange auf eine erfolgreiche Assoziation mit einem Accesspoint gewartet werden soll. Die Zeit für eine erfolgreiche Assoziation hängt stark vom Treiber, von der Karte und vom Accesspoint ab. Es gibt also keinen allgemeinen Wert, der für alle Einstellungen gilt. Ist dieser Wert zu klein eingestellt, kann es passieren, dass der Verbindungsaufbau frühzeitig abgebrochen wird. Ist der Wert allerdings zu lange eingestellt, kann es sein, dass während der Zeit, in der vergebens versucht wird auf einen Accesspoint zu verbinden, ein anderer, besserer Accesspoint verpasst wird.

- autoassoc
  Kann sich die Karte nicht mit einem bestimmten Accesspoint assoziieren, kann der Treiber angewiesen werden, sich mit irgend einem beliebigen Accesspoint in der Nähe zu verbinden. Mit etwas Glück ist so trotzdem eine Verbindung möglich, der Warsurfer Verbindungsalgorithmus wird damit aber bis zu einem gewissen Grad ausgehebelt. Diese Option sollte nur mit Karten benutzt werden, die nicht befriedigend auf direkte Assoziationsrequests reagieren.

- detectorinterface
  Dieser Wert bezeichnet die Karte, mit welcher Netze detektiert werden. Es ist die Karte, die auch in Kismet konfiguriert worden ist.

- connectorinterface
  Dieser Wert bezeichnet die Karte, mit welcher versucht wird eine Verbindung aufzubauen.

- pinghost
  Um festzustellen ob eine Verbindung ins Internet besteht, wird periodisch

ein Host angepingt. Hier können Sie einstellen, welcher Host das ist. Es ist möglich und ratsam mehrere, durch Komma getrennte IP-Adressen anzugeben.

Bekannte Netzwerke

Bekannte Netzwerke können vom Warsurfer besonders behandelt werden, z.B. um kostenpflichtige Hostspots aus dem Verbindungsalgorithmus auszuklammern. Dazu müssen diese in die Tabelle "userconfig" eingetragen werden, mit der "usable"-Spalte auf 0 gesetzt. Beispiel:

```
INSERT INTO userconfig(ESSID,usable) VALUES('WucherGMBH',0);
```

8.5 Anwendung

8.5.1 Zusammengefasster Start

Der Warsurfer besteht aus vielen Applikationen, die einzeln oder zusammen gestartet werden können. Wenn Sie nur eine einzige Konsole bevorzugen, können alle benötigten Applikationen mit dem Befehl "warsurfer" starten. Alle Prozesse ausser dem "Connector" werden dann im Hintergrund versteckt. In diesem Modus ist es aber schwer, allfällige Fehler in den Subprozessen zu erkennen.

8.5.2 Einzelstart

Wenn Sie ein Fan von vielen offenen Konsolen sind, empfehlen wir diese Art, den Warsurfer zu starten. So kann jeder Prozess beobachtet und überwacht werden (dies beeindruckt auch allfällige Zuschauer!). Um die folgenden Befehle auszuführen, müssen Sie in der jeweiligen Konsole in das Installationsverzeichnis wechseln.

1. kismet
2. python detector.py
3. python connector.py
4. python downloaddaemon.py # (nur wenn Sie den Mailfetcher einsetzen oder grössere Downloads planen)
5. python mailfetcher.py # (nur wenn Sie den integrierte Mailfetcher verwenden möchten)

Die Ausgaben der einzelnen Fenster seien hier kurz erklärt:

Kismet

Das Kismetfenster zeigt die vom Sniffer entdeckten Netzwerke. Diese können Sie hier sortieren, genauer untersuchen etc. Für mehr Informationen lese Sie bitte die Kismet Dokumentation.

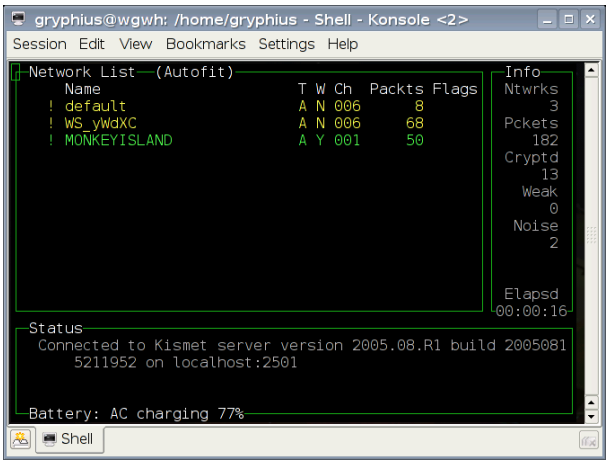


Abbildung 8.7: Kismet Screenshot

Detector

Im Detectorfenster können Sie zusehen, wie die Daten von Kismet in die Datenbank übertragen werden. Jede Zeile zeigt dabei die ESSID, die BSSID sowie allfällig von Kismet erkannte IP-Range Informationen. Zeilen, welche mit NEW: beginnen, bedeuten, dass ein Netzwerk zum ersten mal gesehen wurde. UPD: bedeutet, dass die Daten für ein Netzwerk aktualisiert wurden.

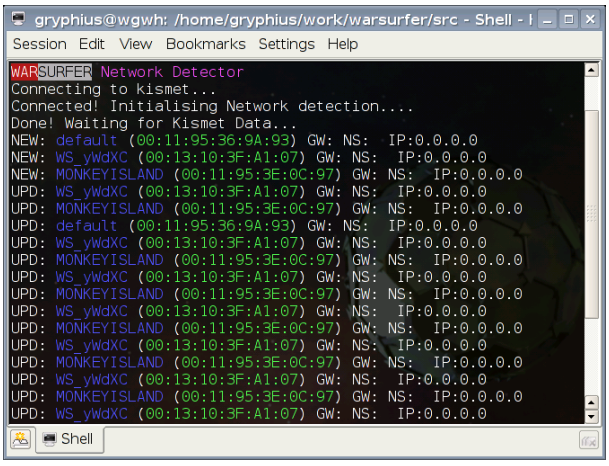


Abbildung 8.8: Detector Screenshot

Connector

Der Connector ist sicherlich die spannendste Applikation des Warsurfers. Er zeigt, mit welchen Netzen der Warsurfer die Verbindung versucht, und v.a. wann

es funktioniert. Diverser Output erfreut hier das Herz des Geeks:

- DB Suggested Net... - Ein möglicher Kandidat wurde ausgewählt, mit dem nun die Verbindung versucht wird.
- Test type: ... - Die Art des Verbindungstest, die versucht wird (DHCP, Fixe IP, Bekannte Herstellersettings etc)
- Associating... - Die Assoziation mit dem Netz wird versucht.
- Modified Settings ... - Gewisse Tests erfordern eine nachträgliche Anpassung der von der Datenbank vorgeschlagenen IP-Settings. Z.b. wenn ein DHCP Request nötig ist, oder wenn der Default Gateway nicht bekannt ist.
- Setting... - Netzwerksettings werden dem System übergeben
- Waiting for available network... - Es ist momentan kein Kandidat in Reichweite
- Connected to ... Ready to rumble! - Die Nachricht , die das Herz höher springen lässt - Sie sind verbunden!

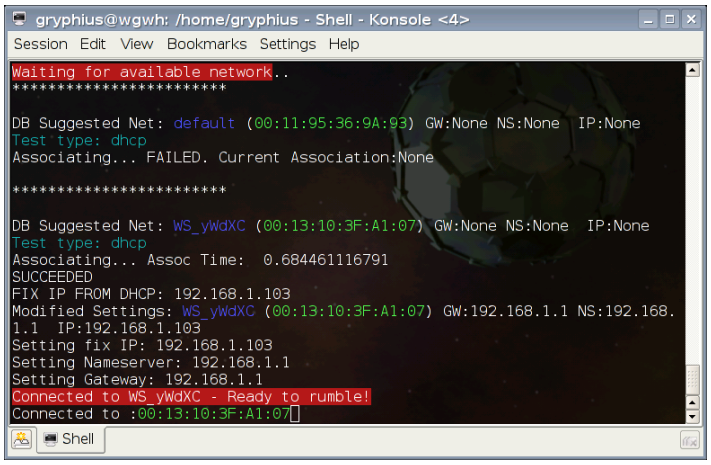


Abbildung 8.9: Connector Screenshot

Download Daemon

Der Download Daemon ist dafür Zuständig, grössere Dateien herunterzuladen, die evtl. nicht an einem Stück transferiert werden können. In diesem Fenster können Sie mitverfolgen, welche Teilstücke gerade in Arbeit sind.

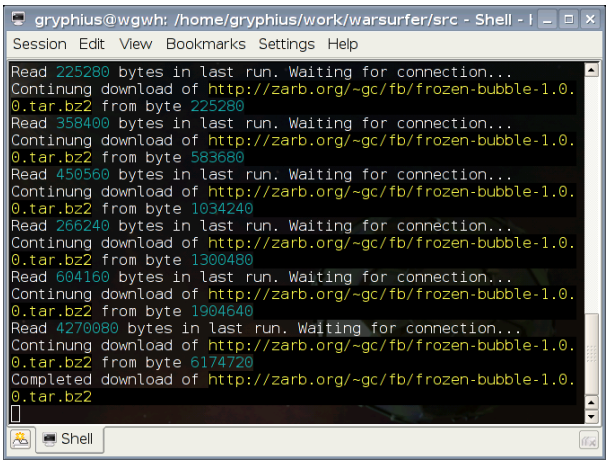


Abbildung 8.10: Download Daemon Screenshot

Mailfetcher

Der Mailfetcher checkt periodisch ob das Indexfile auf dem Server geändert hat. er informiert darüber, wann er das Indexfile zum Download in die Datenbank geschrieben hat. Hat sich das Indexfile seit dem letzten mal verändert, werden die neuen Emailnachrichten zum Download in die Datenbank geschrieben und zugestellt, sobald diese vollständig heruntergeladen worden sind. Ist eine Nachricht zugestellt, gibt der Mailfetcher eine entsprechende Nachricht aus.

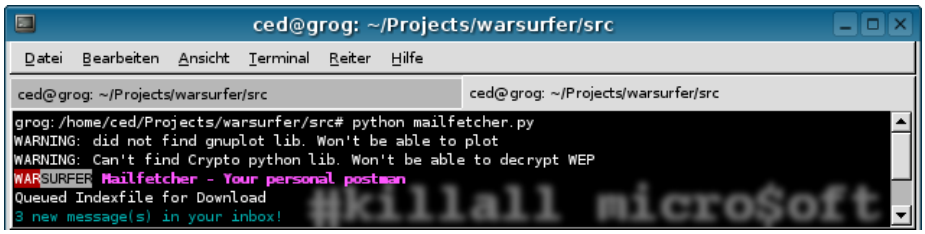


Abbildung 8.11: Mailfetcher Screenshot

8.6 Download grösserer Dateien

Wie sie sicher während der Fahrt schnell feststellen werden, ist es praktisch unmöglich, grössere Dateien im Client-Only Modus herunterzuladen, da die Verbindung zu schnell wieder unterbrochen wird. Zu diesem Zweck beinhaltet der Warsurfer einen Download Daemon, der die Dateien in Einzelstücken herunterladen kann und am Schluss wieder zusammensetzt. Die Aufträge können dem Daemon mit dem Kommandozeilen Tool wsdownload.py (Debian: /usr/-bin/wsdownload) übergeben werden.

```
python wsdownload.py <URL> [-cont]
```

Wird -cont angegeben, wird ein allfällig früher gestarteter Download dieser URL nicht neu gestartet.

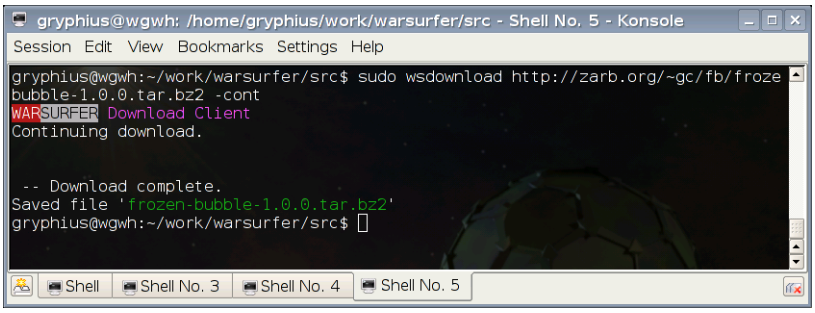


Abbildung 8.12: Download Client Screenshot

8.7 Mailempfang

Eine der Hauptanforderungen des Warsurfers war es, das Emails während der Fahrt heruntergeladen werden.Grundsätzlich kann dies mit dem weiter unten beschriebenen Openvpn sehr einfach gehandhabt werden, indem der Home-Agent einfach die Rolle des Mailservers übernimmt. In diesem Fall sind keine weiteren Konfigurationen notwendig. Natürlich hat nicht jeder einen Server zuhause, und daher gibt es auch die Möglichkeit, den Mailtransfer über den eigenen Webhoster abzuwickeln, sofern dieser einen Shell-Account mit Python bietet (wie z.B. matrixx.ch). In einer späteren Version ist geplant, eine PHP Version des Serverskripts anzubieten, denn PHP wird mittlerweile von praktisch allen Hostern unterstützt. Das Serverskript hat die Aufgabe, die Mails vom POP3 oder IMAP Server entgegenzunehmen und in ein Webverzeichnis abzulegen, wo diese danach vom Warsurfer mittels HTTP geholt werden können. Dabei gelten folgende Konditionen: Jede Mailnachricht gibt eine eigene Datei auf dem Server. Die Datei wird gzip komprimiert und so benannt wie der MD5-Wert ihres unkomprimierten Inhalts plus die Dateiendung .gz. Beispiel: 03f21b939e022176e87feb7704477bbc.gz . Damit der Warsurfer weiss, welche Nachrichten vorhanden sind wird zusätzlich eine Indexdatei angelegt, welche die Namen und Grössen aller Nachrichten auflistet. Diese wird gemäss folgendem Schema generiert:

```
<md5 summe><tab><jobtyp><tab><Grösse in Bytes><URL>
```

Beispiel:

```
07eb51d640b9d9ef345c300eb0101bb8        mail    287     http://ut.darktech.org/
    warsurfer/07eb51d640b9d9ef345c300eb0101bb8.gz
d494cb6b59adfbc9fc4eec39dceacc85        mail    295     http://ut.darktech.org/
    warsurfer/d494cb6b59adfbc9fc4eec39dceacc85.gz
c18a72ab35382761abdcf707fb284075        mail    297     http://ut.darktech.org/
    warsurfer/c18a72ab35382761abdcf707fb284075.gz
```

Das Serverskript ist sehr kurz und kann daher einfach auf die Gegebenheiten Ihres Servers angepasst werden. Hier ein Beispiel des Pythonskripts für IMAP SSL Server:

```
import imaplib,gzip,md5,os
#die URL, die der warsurfer benutzen kann, um die Dateien
    herunterzuladen
baseurl="http://meinhost.com/warsurfer/"
#pfad auf dem Webserver, in dem die Dateien liegen
basepath="/var/www/warsurfer/"
#adresse des mailservers
mailserver='127.0.0.1'
#mailserver benutzername und passwort
username="warsurfer"
password="strenggeheim"

M = imaplib.IMAP4_SSL(host=mailserver)
M.login(username,password)
M.select()
typ, data = M.search(None, 'ALL')
indexfile=gzip.GzipFile(basepath+"index.gz","wb")
for num in data[0].split():
        typ, msgdata = M.fetch(num, '(RFC822)')
        msg=msgdata[0][1]
        md5sum=md5.new(msg).hexdigest()
        filename="%s.gz"%md5sum
        file=gzip.GzipFile(basepath+filename,"wb")
        file.write(msg)
        file.close()
        size=os.path.getsize(basepath+filename)
        indexline="%s\t%s\t%s\t%s"%(md5sum,"mail",size,
            baseurl+filename)
        indexfile.write(indexline+"\n")
indexfile.close()
M.close()
M.logout()
```

Dieses Skript können Sie nun per Cronjob periodisch ausführen lassen (z.b. alle 20 Minuten). Das Webverzeichnis sollten Sie nun zusätzlich mittels .htaccess/.htpasswd schützen, ausser Sie möchten der Welt ihre privaten Emails zur Verfügung stellen.
Konfigurieren sie nun die Einstellungen in der Warsurferdatenbank (mailindexfile,mailhttpuser,mailhttppassword,mailfile), sowie evtl. ihre Emailapplikation, dass diese ihr lokales Mailfile anzeigen kann.

8.8 Surfen mit Openvpn

Um das Surfen mit dem warsurfer einigermassen erträglich zu gestalten, empfehlen wir dringend die Installation der Software OpenVPN. Dafür ist (wie für den Mailfetcher) ein Home-Agent nötig. Mittels OpenVPN wird, sobald der Warsurfer während der Fahrt eine Verbindung ins Internet herstellen kann, eine VPN Verbindung zum Home-Agent erstellt, welche es ermöglicht, TCP-Verbindungen über mehrere Accesspoints hinweg am Leben zu erhalten. Der Browser kann so z.b. Websites laden, ohne das Sie genau den Moment abpassen müssen, in dem

Sie mit dem Internet verbunden sind. Auch SSH Sitzungen, Chats etc. können so einen Wechsel von Netzwerk zu Netzwerk überleben, vorausgesetzt der Unterbruch dauert nicht zu lange.

8.8.1 Installation

Debian Benutzer sind hier wieder mit einem simplen

```
apt-get install openvpn
```

bedient. Benützer anderer Distributionen verwenden die mitgelieferten Pakete oder installieren wie gehabt ab aus den Sourcen, welche auf folgender Webseite erhältlich sind: http://openvpn.net/download.html . Zusätzlich zu OpenVPN sollte auf dem Homeagent ein HTTP Proxy installiert werden. Ein intelligenter Proxy, welcher Link Prefetching unterstützt ist hier von Vorteil.

8.8.2 Konfiguration

Als erstes muss ein statischer Schlüssel definiert werden. Führen Sie dazu auf dem Home Agent folgendes Kommando aus:

```
openvpn --genkey --secret warsurfer.key
```

Kopieren Sie den generierten Schlüssel auf dem Home-Agent und auf dem Notebook in das Verzeichnis /etc/openvpn

Erstellen Sie auf dem Server eine Konfiguration warsurfer_server.conf mit folgendem Inhalt:

```
dev tun

# 10.1.0.1 is our local VPN endpoint (office).
# 10.1.0.2 is our remote VPN endpoint (home).
ifconfig 10.1.0.1 10.1.0.2

# Our up script will establish routes
# once the VPN is alive.
up ./warsurfer_server.up

# Our pre-shared static key
secret warsurfer.key

port 1337

comp-lzo


# Uncomment this section for a more reliable detection when a system
# loses its connection.  For example, dial-ups or laptops that
# travel to other locations
ping 15
ping-restart 45
```

```
ping-timer-rem
persist-tun
persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

Damit nach erfolgreicher Verbindung die korrekte Route zum Warsurfer gesetzt wird, muss noch folgendes Skript mit Namen warsurfer_server.up im selben Verzeichnis erstellt werden:

```
#!/bin/sh
route add -net 10.0.1.0 netmask 255.255.255.0 gw $5
```

Befindet sich der Home-Agent hinter einem NAT Gerät, muss dort der Port 1337 weitergeleitet werden. Der OpenVPN Server wird folgendermassen gestartet:

```
openvpn --config warsurfer_server.conf
```

Der HTTP Proxy muss evtl. noch konfiguriert werden, dass er Verbindungen von 10.0.1.2 (Warsurfer) zulässt.

Auf dem Warsurfer sieht die Konfiguration warsurfer_client.conf folgendermassen aus:

```
dev tun

# Our OpenVPN peer is the office gateway.
remote myhomeagent.com

# 10.1.0.2 is our local VPN endpoint (home).
# 10.1.0.1 is our remote VPN endpoint (office).
ifconfig 10.1.0.2 10.1.0.1

# Our up script will establish routes
# once the VPN is alive.
up ./warsurfer_client.up

# Our pre-shared static key
secret warsurfer.key

port 1337

comp-lzo

# Uncomment this section for a more reliable detection when a system
# loses its connection.  For example, dial-ups or laptops that
# travel to other locations.
```

```
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key

# Verbosity level.
# 0 -- quiet except for fatal errors.
# 1 -- mostly quiet, but display non-fatal network errors.
# 3 -- medium output, good for normal operation.
# 9 -- verbose, good for troubleshooting
verb 3
```

myhomeagent.com muss dabei mit dem Hostnamen des Home-Agents ersetzt werden. Wenn möglich, sollte hier sogar die IP-Adresse eingesetzt werden, da in diesem Fall OpenVPN nicht zuerst einen langsamen DNS-Lookup durchführen muss. Auch auf dem Client ist ein Routing Skript nötig (warsurfer_client.up):

```
#!/bin/sh
route add -net 10.0.0.0 netmask 255.255.255.0 gw $5
```

Der Client kann nun während einem Ride genau gleich gestartet werden:

```
openvpn --config warsurfer_client.conf
```

Im Browser sollte nun als Proxy der VPN Server (10.1.0.1) eingetragen werden.

8.9 Tweaks

Diverse Tricks verbessern die Funktionalität des Warsurfers bezüglich einzelner Anwendungen. Einige Beispiele sind hier zusammengetragen.

8.9.1 Firefox

Der Firefoxbrowser kann mit diversen Tricks (Link Prefetching, HTTP Pipelining, DNS Cache etc.) dazu gebracht werden, die zur Verfügung stehende Bandbreite optimal zu nutzen. Diese Tricks werden in der [14] Fasterfox Extension zusammengefasst, welche auf folgender Webseite erhältich ist: http://fasterfox.mozdev.org/

8.9.2 SSH

SSH Sitzungen können sehr lange Unterbrücke überleben, wenn in der Konfiguration die Option "TCPKeepAlive No" gesetzt wird. Achtung: Kann eine Sitzung nicht wieder hergestellt werden, muss diese manuell auf dem Server gestoppt werden!

8.9.3 DNS

Viel Zeit geht durch DNS Anfragen verloren. Wir empfehlen daher, einen lokalen DNS Proxy zu installieren, welcher die IP-Adressen Ihrer favorisierten Webseiten speichert. Es ist geplant, in einer späteren Version des Warsurfers eine direkte Konfigurationsmöglichkeit für DNS Proxies einzubauen.

Kapitel 9

Tests

Mit der in der Analyse beschriebenen Accesspoint-API sind hier einige automatisierte Tests definiert worden, die die Funktionalität des Warsurfers überprüfen sollen.

9.1 Netzerkennung

Testbeschreibung Dieser simple Test generiert 10 neue Netzwerke mit neuen BSSIDs , wobei immer nur ein Netz gleichzeitig aktiv ist. Die Netze sind unterschiedlich lang verfügbar und bieten eine Verbindung ins Internet. Es soll getestet werden, ob der Warsurfer diese Netze korrekt erkennt und in die Datenbank einträgt.

Testskript

1. wstest_random1.py - Erzeugt Netze
2. wstest_detector.py - Liefert die Resultate

Benötigte Infrastruktur

- Acess Point Linksys WRT54g mit OpenWRT Firmware
- Eine Wirelesskarte als Detector

Durchführung

1. Warsurfer Datenbank leeren (kismetdata, userconfig, connectsettings)
2. Kismet starten
3. Detector starten
4. Testskript 1 laufen lassen, warten bis es beendet ist.
5. Resultate mittels Testskript 2 überprüfen

Erfolgsbedingungen

1. 30 von 30 Netzen wurden in kismetdata eingetragen
2. 30 von 30 Netzen wurden in availlog eingetragen

Durchführungen

Durchführung mit Linksys WPC11

```
Detector Test running at:  Mon Nov 28 13:56:51 2005
Success condition 1:  10 entries in kismetdata... PASSED
Success condition 2:  10 unique entries in availlog... PASSED

OVERALL TEST RESULT:
**********
* PASSED *
**********
```

Durchführung mit Cisco Aironet 350

```
Detector Test running at:  Mon Nov 28 14:07:09 2005
Success condition 1:  10 entries in kismetdata... FAILED (1)
Success condition 2:  10 unique entries in availlog... FAILED (1)

OVERALL TEST RESULT:
**********
* FAILED *
**********
```

Durchführung mit USR 5410

Bemerkung: Dieser Test wurde mit aktiviertem Warsurfer-Channelhopper (6,11,1,7,8,3,2,5,4,9,10, Freq: 0.1 sec) durchgeführt.

```
Detector Test running at:  Mon Nov 28 14:22:54 2005
Success condition 1:  10 entries in kismetdata... PASSED
Success condition 2:  10 unique entries in availlog... PASSED

OVERALL TEST RESULT:
**********
* PASSED *
**********
```

Diskussion der Resultate

Grundsätzlich funktioniert die Detektion, sie hängt aber von der verwendeten Hardware ab. Die Cisco-Karte bliebt auf einem Kanal mit einem verschlüsselten Netzwerk hängen und wechselte nicht mehr, daher wurde nur ein Netz gefunden.

9.2 Verbindung

Testbeschreibung Ziel Dieses Tests ist es, herauszufinden, wie gut der Verbindungsalgorithmus des Warsurfer funktioniert. Dazu werden 10 aufeinander folgende Netze generiert, die einer Verbindung ins Internet ermöglichen. Jedes Netz ist genau 10 Sekunden verfügbar.

Testskript

1. wstest_random1_hostap_connector.py - Erzeugt Netze
2. wstest_connector.py - Liefert die Resultate

Benötigte Infrastruktur

- Laptop mit Wirelesskarte (HostAp Treiber)
- Zwei Wirelesskarten (Detector, Connector)

Durchführung

1. Warsurfer Datenbank leeren (kismetdata, userconfig, connecttries)
2. kismet starten
3. Detector starten
4. Connector starten
5. Testskript 1 laufen lassen
6. Resultate mit Testskript 2 prüfen

Erfolgsbedingungen

1. 8 von 10 Netzen wurden erfolgreich in connectlog eingetragen
2. Die durchschnittliche Onlinezeit beträgt mindestens 3. Sekunden.

Durchführung

```
Connector Test running at:  Tue Dec 13 12:15:45 2005
Success condition 1:  8 entries in connectlog... PASSED (8)
Success condition 2:  minimum 3.0 seconds average internet connection time
 PASSED (7.266097)

OVERALL TEST RESULT:
**********
* PASSED *
**********
```

9.3 Maildownload

Testbeschreibung Dieser Test soll feststellen, ob der Mailtransfer über den Warsurfer funktioniert. Dazu wird ein Mail mit einem ca. 5MB Attachment an die Adresse warsurfer@ut.darktech.org geschickt. Dort läuft das Mailserver Skript, welches die Daten für den HTTP Transfer bereitstellt. Auf der Clientseite wird wieder das gleiche Setup wie beim letzten Test verwendet (10 Netze à 10 Sekunden). Der Test gilt als bestanden, wenn das Mail in dieser Zeit transferiert wurde und die MD5 Summe des Attachments übereinstimmt. Die Attachment wurde bewusst überdurchschnittlich gross definiert, da in unserem Testsetup auch eine überdurchschnittlich schnelle Leitung zum Mailserver besteht.

Testskript wstest_random1_hostap_connector.py - Erzeugt Netze

Benötigte Infrastruktur

- Laptop mit Wirelesskarte (HostAp Treiber)
- Zwei Wirelesskarten (Detector, Connector)

Durchführung

1. Warsurfer Datenbank leeren (kismetdata, userconfig, connecttries)
2. Kismet starten
3. Detector starten
4. Connector starten
5. Mailnachricht an warsurfer@ut.darktech.org senden (md5 summe prüfen)
6. Testskript laufen lassen
7. nach Ablauf die Inbox überprüfen

Erfolgsbedingungen

1. Mail Nachricht wurde in das Lokale Mailfile eingetragen
2. MD5 Summe des Anhangs stimmt

Durchführung

13. Dez 12:43
Der Mailanhang wurde mit folgendem Kommando generiert:

```
dd if=/dev/urandom of=5mdata count=5 bs=1M
```

MD5 Summe des Mailanhangs (md5sum 5mdata) : d5af7eed4d7b950138e1bc5fe1fc53b5
Da für diesen Test kein automatisches Skript besteht, hier die Ouputs der involvierten Applikationen:

Download Daemon:

```
WARSURFER DOWNLOAD DAEMON UP AND RUNNING!
Starting download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/index.gz
Completed download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 index.gz
downloaddaemon.py:194: RuntimeWarning: tmpnam is a potential security
risk to your program filelocation=os.tmpnam()
Starting download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz
Read 665600 bytes in last run. Waiting for connection...
Continung download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz from byte 665600
Error while downloading  http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz (DNS error).
Continung download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz from byte 665600
Error while downloading  http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz (DNS error).
Continung download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz from byte 665600
Completed download of http://warsurfer:stawsa@ut.darktech.org/warsurfer/
 07f5d5b23999d67b0279a55b74122aed.gz
```

Mailfetcher:

```
WARSURFER Mailfetcher - Your personal postman
Queued Indexfile for Download
1 new message(s) in your inbox!
```

Resultat: Die Mailnachricht ist erfolgreich angekommen, die MD5 Summe des Anhangs stimmte überen.

9.4 Feldtest

Natürlich muss sich der Warsurfer unter realen Bedingungen unter Beweis stellen. Zu diesem Zweck haben wir mehrere Stunden in in Trams, Zügen und Bussen in und um Zürich verbracht. Dabei wurden sämtliche Features getestet, d.h. Client-Only Modus, Mailfetcher mit Home Agent, Roaming mit OpenVPN etc.

Feldtest 1, 1.12.2005

Strecken: Tramlinien 2,3,6,7,8,9,10 von Endstation bis Endstation., Buslinie 80 (Albisrieden bis Örlikon)

Getestet wurden: Client-Only Modus (direkter Download), Mailfetcher und geroamtes Surfen über OpenVPN.

Hardware: Setup 1: Fujitsu Siemens Amilo A7640, Detector: Linksys WPC 11 v3(hostap), Connector: Surecom EP-9001 (zd1201)
Setup 2: IBM Thinkpad T30, Detector: Linksys WPC 11 v3(hostap), Connector: Intel Wireless Pro miniPCI (hostap)

Resultate

Bei den Folgenden Angaben wurden nur private Netze mit einbezogen (also keine "Public Hotspots")

Setup 1

- Total entdeckte Netzwerke: 1262
- Davon ungeschützt: 438
- WEP: 633
- Anderer Schutz (WPA etc.): 191
- Erreichte Verbindungen während der Fahrt / an Haltestellen: 61 Verbindungen auf 32 verschieden Netzwerken
- Die durchschnittliche Verbindungsdauer lagen bei 5.3 Sekunden

Setup 2

- Total entdeckte Netzwerke: 437
- Davon ungeschützt: 159
- WEP: 229
- Anderer Schutz (WPA etc.): 49
- Erreichte Verbindungen während der Fahrt / an Haltestellen: 11 Verbindungen auf 8 verschieden Netzwerken
- Die durchschnittliche Verbindungsdauer lag bei 7.5 Sekunden

Diskussion

Statistisch gesehen war die Ausbeute eher gering, Nur 5 bzw. 7 Prozent der offenen Netzwerke konnten wirklich verwertet werden. Trotzdem ist der Test erfolgreich verlaufen, denn: Mails konnten gelesen werden, Surfen im Tram war mit etwas Geduld ebenfalls möglich. Beim zweiten Setup wurden weitaus weniger Netzwerke gefunden, da hier einerseits die Connectorkarte viel schwächer war, andererseits der Ride aus batterietechnischen Gründen nicht so lange dauerte.

Teil IV

Bewertung und Ausblick

Kapitel 10

Resultate und Zielerreichung

10.1 Zielerreichung gemäss Aufgabenstellung

Entwicklung einer Testumgebung Dieses Ziel wurde erreicht. Mittels den [15] HostAp-Treibern auf dem Notebook und mit dem Linksys WRT Accesspoint konnten wir per Skripts beliebig Wireless-Netzwerke mit verschiedenen BSSIDs/ESSIDs erzeugen, um so eine Fahrt zu simulieren. Ein standardisiertes Python Interface erlaubt die einfache Erweiterung auf andere Arten von konfigurierbaren Accesspoints, so dass z.b. auch mehrere Netze gleichzeitig simuliert werden können.

Entwicklung einer Applikation, die sich laufend mit offenen Netzwerken verbindet. Dieses Ziel wurde erreicht. Der Warsurfer "Connector" sucht sich automatisch das beste verfügbare Netzwerk und verbindet sich damit. Der Erfolgsfaktor hängt hier aber sehr stark von der verwendeten Hardware und natürlich von der gefahrenen Strecke ab. Der Feldtest in Zürich zeigte, dass von allen erkannten ungeschützten Netzen nur ca. 7% wirklich verwendet werden konnten, hier gibt es sicherlich noch Verbesserungspotenzial.

Eigenentwicklung oder Implementation eines Roaming-Protokolls, welches zumindest das Lesen von Email-Nachrichten erlaubt. Dieses Ziel wurde erreicht. Dank dem Warsurfer "mailfetcher" können auch grössere Maildateien heruntergeladen werden. Dafür ist aber ein speziell Konfigurierter Server nötig. Mittels der Software OpenVPN ist sogar vollständiges Roaming möglich, d.h. , Verbindungen werden auch über mehrere Netzwerkwechsel hinweg nicht abgebrochen. Dies erlaubt das Arbeiten mit SSH Sessions etc. Aber auch hier muss man sich bewusst sein, dass das Surfen auf diese Art viel Geduld erfordert. Oftmals sind Netze nur so kurz Verfügbar, dass die Zeit nicht reicht, eine Verbindung herzustellen.

Fazit Gemäss Aufgabenstellung wurden alle Ziele erreicht. Es ist tatsächlich möglich, seine Emails während einer Busfahrt herunter zu laden und zu lesen.

Ob es wirklich funktioniert hängt aber einerseits stark von der Länge der Busfahrt und andererseits vom Gebiet in dem man sich bewegt ab.
Im Internet zu surfen erfordert ein grosses Mass an Geduld und der Erfolg ist eng mit der Reaktionsfähigkeit des Anwenders verbunden, weil dieser ein Auge auf den Connector haben muss und seinen Request mit Vorteil erst dann absetzt, sobald die Verbindung aufgebaut ist, um so das Risiko eines Timeouts des Browsers möglichst gering zu halten.
Die Installation des Warsurfers ist relativ umständlich und somit für Anfänger ungeeignet. Ein Teil der Software befindet sich im Status "unstable". Nach der erfolgreichen Installation müssen an verschiedenen Orten Konfigurationsdateien angepasst werden, was das Risiko einer Fehlkonfiguration erhöht.
Abschliessend lässt sich sagen, dass der Aufwand der Installation nicht im Masse mit dem Ertrag bei der Anwendung steht. Die Software richtet sich in diesem Status noch eher an Internetsüchtige Geeks mit hohem Mass an Experimentierfreudigkeit.

10.2 Feldtest

Während unseren Testfahrten in und um Zürich haben wir einiges Datenmaterial gesammelt. Insgesamt sind wir an 4824 Accesspoints vorbeigefahren. Davon waren 551 Kostenpflichtige Public-Hotspots (tpn.ch, Mobile, Connectionpoint etc.). Es zeigt sich, dass sich das Sicherheitsbewusstsein der Bevölkerung in den letzten Jahren doch etwas gebessert hat: Von den 4273 nicht öffentlichen Hotspots waren 2012 (48%) mit WEP, 491 (11%) mit anderen Verschlüsselungsarten wie WPA gesichert. Es bleiben noch 1770 (41%) unverschlüsselte Netzwerke. Die ESSIDs werden meistens nicht geändert, wie folgende Hall-Of-Fame zeigt:

1. keine ESSID : 374
2. default : 235
3. NETGEAR : 113
4. Wireless : 95
5. linksys : 87
6. USR8054 : 30
7. USR5462 : 17

Werden ESSIDs geändert, dann meist auf den eigenen Namen, die Firma oder die Strasse. Aber auch einige kreative ESSIDs konnten wir entdecken:

- Bea(t)s ExcessPoint
- CiaoFabyClaudiaLuschtUfeGlasWii
- Em Fritz sis WLAN
- fingerweg
- Hasebär :)

- heinis-we-lan
- hitchhicker
- Houston Controll
- mach Dich vom Acker!
- no access baby

10.3 Soll-Ist Vergleich

Geschätzter Aufwand in Stunden: 340
Tatsächlicher Aufwand in Stunden: 372
Die Differenz der beiden Stundenzahlen hat vor allem den Grund, dass Anfangs grosse Probleme mit den Treibern herrschten, welche erst bewältigt werden mussten. Dies bedeutete einen erheblichen Mehraufwand, welcher aber durch den Einsatz von OpenVPN anstatt einer Implementation eigener Software wieder wett gemacht werden konnte.
Tatsächlich war geplant nach der ersten Iteration einen Prototypen zur Verfügung zu haben, welcher sich bereits automatisch mit Netzen verbinden kann. Dies war aufgrund des erwähnten Problems mit den Treibern nicht möglich. Der Connector war nach der ersten Iteration nicht fähig sich zuverlässig mit verschiedenen Netzen zu verbinden. So konnten lediglich die Werte welcher der Detector geliefert hat als Referenz für die zweite Iteration, in der es darum ging das Roaming zu implementieren, genommen werden. Anfangs der zweiten Iteration wurde das Treiberproblem durch den Einsatz neuer Hardware gelöst und der getroffene Entscheid OpenVPN einzusetzen konnte unter realen Bedingungen getestet werden. Die Resultate übertrafen die Erwartungen. Deshalb wurde beschlossen auf die aufwändige Variante das Roaming selbst zu programmieren zu Gunsten weiterer Verfeinerungen im Connector fallen gelassen.
Gegen Ende des Projektes ging es darum die Dokumentation termingerecht fertigzustellen was ebenfalls ein paar Überstunden mit sich brachte.

Kapitel 11

Persönliche Eindrücke

11.1 C. Wider

Die Idee, während einer Busfahrt über vorbeifahrende Drahtlose Netzwerke eine quasi stetige Verbindung ins Internet zu erhalten hat mich stark fasziniert. Sofort war ich mit grossem Elan dabei als es darum ging erste Codezeilen zu schreiben und damit verbunden auch Tests zu machen, deren Resultate die anfängliche Euphorie eher geschürt als gedämpft hatten. Umso grösser war die Freude als sich Herr Dr. Steffen dazu begeistern liess, unsere Arbeit als Dozent zu betreuen.
Als sich während der Diplomarbeit aber zeigte, wie sehr eine erfolgreiche Verbindung mit einem Accesspoint in nützlicher Frist von der darunterliegenden Hardware abhängt, machte sich Enttäuschung breit. Wie können die Wireless Treiber unter Linux so schlecht implementiert sein, dass teilweise Befehle, die auf die Karte abgesetzt werden einfach ignoriert werden? - Schliesslich fanden wir aber eine Hardwarekonfiguration die vielversprechend war. Und nach einigen Erkenntnissen und Verbesserungen des Codes war es tatsächlich möglich, während einer Busfahrt Emails herunterzuladen.
Ich persönlich bin mit dem Warsurfer, so wie er jetzt ist, nur bedingt zufrieden. Er erfüllt zwar alle Anforderungen der Aufgabenstellung, von quasi stetiger Verbindung kann aber, bei einer Ausbeutung von 7% aller offenen Netze, nicht die Rede sein. Bestimmt werde ich nach Abschluss der Diplomarbeit einiges an Freizeit darin inverstieren das Surferlebnis und die Netzausbeute erheblich zu verbessern.
Mit O. Schacher zusammen zu arbeiten war, auch in diesem Projekt, sehr angenehm. Während dieser Arbeit trug seine Fähigkeit zum analytischen Denken genauso wie sein tiefgründiges Wissen über sämtliche Bereiche welche entweder mit dem Internet oder mit Linux zu tun haben massgeblich zum Erfolg des Projektes bei. Vor allem aber ist es seine Fähigkeit abstrakte Gedanken in ausführbaren Code umzusetzen, seine Art zu kommunizieren und seine Begeisterungsfähigkeit, was die Zusammenarbeit mit ihm zum Vergnügen macht.
Die Arbeit am Warsurfer hat mich einiges im Umgang mit Datenbanken gelehrt. Ebenso wurden meine Python-skills verbessert. In diesem Projekt war die Testumgebung besonders wichtig. Die Arbeit an diesem Projekt hat meine Fähigkeit, das Bild der realen Welt auf das für eine Applikation wesentliche zu

reduzieren und nachzubauen, geschult.

11.2 O. Schacher

Bis zu einem gewissen Grad bin ich Internetsüchtig. Wenn ich irgendwo in die Ferien gehe (nicht, dass dies oft vorkäme), wird als erstes mal das Notebook ausgepackt und Kismet angeworfen, um nachzuschauen, ob mir ein freundlicher Nachbar vielleicht die Möglichkeit bietet über Instant Messaging die daheimgebliebenen auszulachen. Als Cédric mit der Idee auf mich zukam, dass man eventuell eine Internetverbindung sogar während einer Busfahrt herstellen könnte, war ich natürlich sofort davon begeistert und konnte es kaum erwarten, mit diesem Projekt zu beginnen. Das sich daraus sogar eine Diplomarbeit entwickelt hat, freut mich besonders. Ich hatte die ganze Zeit über Spass an der Arbeit, auch wenn diese einige Nachtschichten erfordert hat, um Probleme zu lösen, die von meist anderen verursacht wurden (Wireless Treiber unter Linux.... *seufz*). Die Zusammenarbeit hat in meinen Augen wunderbar funktioniert. Ein Dozent der HSR meinte einmal “Wenn das Problem gross ist, ist die Lösung nah”, und genau nach diesem Motto schaffte es Cedi immer wieder, bei auf den ersten Augenschein ausweglosen Situationen die richtige Idee zu liefern. Auch Herrn Steffen möchte ich ganz herzlich für seine Tipps danken, mit ihm als Betreuer haben wirklich einen Glücksgriff getätigt!
Wie Cédric bin auch ich mit dem Resultat noch nicht ganz zufrieden, denn “Internet everywhere” ist noch nicht möglich. Ich freue mich bereits auf die Zeit nach der Diplomarbeit, wo wir uns wieder mehr dem Code und weniger der Dokumentation widmen können ;-). Der Lernfaktor war bei diesem Projekt riesig: Die Arbeit OpenVPN, der Bau von Debian-Paketen und PL/PGSQL Datenbankskripte sind nur einige Beispiele, die ich bestimmt in anderen Projekten weiterverwenden kann.

Kapitel 12

Ausblick

12.1 Verbleibende Probleme

Aus zeitlichen Gründen konnten leider nicht ganz alle Probleme gelöst werden. Diese seien hier aufgelistet.

- DNS Caching
  Während den Testphasen hat sich herausgestellt, dass der "Flaschenhals" oft im DNS-Lookup liegt. Da wir annehmen, dass ein typischer "Warsurfer" meist die selben Seiten anschaut (slashdot.org, digg.com etc.) würde ein lokaler DNS Proxy sicher starke Verbesserungen mit sich bringen. Die Implementation im Warsurfer liesse sich relativ einfach einbauen. Es wäre eine neue Konfigurationsoption nötig, sowie die Anpassung des Codestücks im Connector, welches die Nameserver setzt. Zudem müsste ein solches Setup natürlich dokumentiert werden, da ein neuer Daemon weitere Komplexität bringt.

- Bessere Installationsskripte
  Diverse Verbesserungen an der Installation wären nötig, um diese auch für technisch weniger versierte Benutzer zu ermöglichen, beispielsweise eine automatische Anpassung der Kismet- und Postgreskonfiguration.

- Scapy Fehler
  Scapy schreibt leider öfters ungefragt Warnungen und Fehler auf die Konsole ("Gnuplot not installed", "child died unexpectedly", ...) Es wäre schön, wenn sich die Fehler beheben und die Warnungen verstecken liessen.

- Public Hotspots
  Kostenpflichtige Hotspots werden, soweit bekannt, von der Datenbank herausgefiltert. Trifft der Warsurfer aber auf einen bisher unbekannten Hotspot, glaubt der Warsurfer, er sei mit dem Internet verbunden, da die Syn-Pings auf Port 80 vom Hotspotprovider beantwortet werden.

- Ping detection Es kann vorkommen, dass scapy keine Antwort auf die ICMP- und Synpings meldet, obwohl eine Verbindung ins Internet besteht. Ev könnte man hier zusätzlich den Verbindungsstatus von OpenVPN abfragen. Dieser scheint ziemlich zuverlässig zu funktionieren.

12.2 Weiterentwicklung

Zusätzlich zu den Bugfixes gibt es natürlich noch diverse Weiterentwicklungen, die wir nach der Diplomarbeit in Angriff nehmen werden.

- Verbindungsalgorithmus Momentan können nur ca. 7% der gefundenen offenen Netze verwendet werden. Neben besserer Hardware könnte man sicherlich noch einige Tweaks bei der Verbindung suchen. Optimierung der Polling- und Sleepintervalle, direkter Eingriff in den ARP-Cache, Priorisierung der Accesspoints mit kürzerer Assoziationszeit, ...

- GUI
  Eine Grafische Benutzeroberfläche würde die Anwendung für weniger konsolenbegeisterte Anwender erleichtern. Neben diversen Statusanzeigen (Verfügbare Netze, verbundene Netze, Fortschritt der Downloads, Transferierte Emails, ...) wären sicher ein Konfigurationseditor eine starke Verbesserung in der Benutzungsfreundlichkeit.

- Remote Protokoll
  Das Hauptproblem liegt momentan sicher in der verwendeten Hardware. Eine Idee für den Warsurfer V2 ist, das man einen WRT Accesspoint auf das Autodach montieren kann, um so ein besseres Signal zu erhalten. Der Code müsste so erweitert werden, dass er die Verbindungskommandos anstatt lokal auf dem über IP erreichbaren WRT Router absetzen kann (z.B. über SSH.

Abbildung 12.1: Der zukünftige Warsurfer?

- Nur eine Wirelesskarte
  Momentan muss der Warsurfer mit zwei Wirelesskarten (Detector / Connector) betrieben werden. Eventuell liesse sich hier eine Möglichkeit finden, diese beiden Aufgaben in einer Karte zu vereinen.

- GPS
  Mit einem GPS-Gerät kann man die Position der Netzwerke ziemlich genau bestimmen. Der Lernalgorithmus könnte so erweitert werden, dass er bei einer wiederholten Durchfahrt durch ein Gebiet bessere Netze auswählt (man kann z.b. ein Verfügbares Netz auslassen, weil man weiss, das gleich ein viel besseres kommt)

- Unterstützung von Instant Messaging
  Zur Zeit wird nur das Herunterladen von Emails durch den Warsurfer und Browsen über den OpenVPN Tunnel unterstützt. Eine erwünschte Weiterentwicklung wäre es Instant Messaging zu unterstützen. Es wäre denkbar das Protokoll analog zum Email Protokoll über einen Home Proxy zu realisieren.Zusätzlich zum Home Proxy müsste aber auch auf dem Client eine Proxy Instanz installiert werden, welche dem Instant Messenger eine stetige Verbindung vorgaukelt und diesen mit Nachrichten füttert, sobald diese vorliegen. Der Home Proxy seinerseits übernähme die Kommunikation mit dem Accounting Server, damit der Benutzer für seine Freunde nicht als offline dargestellt wird.

- RSS Newsreader
  Eine ebensogrosse Nachfrage wie nach seinen Emails besteht bestimmt auch nach News. Um den Browsing Verkehr einzuschränken könnten diese, wie die Emailnachrichten, über den Homeproxy geladen werden und dem Benutzer beispielsweise über einen lokalen Webserver zur Verfügung gestellt werden. Das hätte den Vorteil, dass die News, wie gewohnt über den Webbrowser angesehen werden können. Die News würden herunter geladen werden, sobald eine Verbindung besteht ohne dass der Benutzer diese explizit mit einem Browser zum Zeitpunkt einer stehenden Verbindung anfordern müsste.

Kapitel 13

Dank

Eigen Personen und Institutionen möchten wir ganz besonders Danken, denn sie haben massgeblich zum Erfolg des Warsurfers beigetragen:

- Allen voran natürlich Herrn Steffen, unserem Betreuer. Einerseits dafür, dass er sich sofort bereit erklärt hat, unser Projekt zu betreuen, andererseits natürlich für seine wertvollen Tipps welche uns oft aus technischen Sackgassen herausgeführt haben!
- Herrn B. Stettler und Herrn R. Stanger vom ITA dafür, dass sie uns diverse Wirelesskarten zu günstigen Konditionen überlassen haben.
- IBM Schweiz für die generelle Unterstützung z.B. durch Anstellungsbedingungen welche das Schreiben der Diplomarbeit in dieser Zeit ermöglichten
- Der Hochschule Rappperswil für die Möglichkeit eine derartige Diplomarbeit durchzuführen.
- McDonalds/Pizza Dieci, welche auch dieses Semester wieder für unser leibliches Wohl vor den wöchentlichen Sitzungen mit Herrn Steffen gesorgt haben
- Dem Zürcher Verkehrsverbund, der uns während kalten Tagen stets ein warmes "Bänkli" in einem seiner unzähligen Trams zur Verfügung gestellt hat.
- S. Roth für zahlreiche Feinkostmenues, welche die Moral während langer Arbeitstage wieder anzuheben vermochten.
- D. Schlumpf für Rücksichtnahme und Verständnis für die geleistete Mehrarbeit.
- "Miss Handballmannschaft" die uns immer wieder ein Lächeln aufs Gesicht gezeichnet und uns nachhaltig geprägt hat.
- Der grösste Dank aber gilt allen den anständigen Bewohnern der Stadt Zürich, die noch an das Gute im Menschen glauben und ihre Wireless Netzwerke deshalb nicht verschlüsseln.

Teil V

Appendix

Kapitel 14

Risikoanalyse

Die untenstehende Tabelle zeigt die Risiken, welche für diese Arbeit relevant sind und die zu treffenden Massnahmen bei Eintreten des Risikos. Der Gewichtete Schaden bezeichnet die Anzahl Tage, die für dieses Risiko als Reserve eingeplant werden.
Am schwersten wurde der Fall eingeschätzt, wenn das definierte Testgebiet (die Stadt Zürich) zu wenige offene Netze bietet und nach einem neuen Gebiet Ausschau gehalten werden müsste. Ein neues Testgebiet zu suchen ist nicht ganz einfach, da die Suche mit enormer Sniffer-Arbeit verbunden wäre.
Andere Risiken, wie der Ausfall eines Mitarbeiters durch Krankheit hat geringe Folgen. Der Arbeitspartner müsste in diesem Fall etwas mehr Leisten. Trotzdem würde der von Krankheit geplagte Mitarbeiter wohl zu einem grossen Teil im Bett weiterarbeiten.
Sollte es vorkommen, dass während einer Busfahrt die Verbindungszeit zu kurz ist, müsste auf ein langsameres Fortbewegungsmittel umgestiegen werden. Frühe Tests sollen hier Aufschluss über die tatsächlichen Begebenheiten geben, sodass die Testumgebung gemäss den durch diese Tests erlangten Daten aufgesetzt werden kann.
Selbstverständlich wurde auch auf den Ausfall von Arbeitsmitteln (Notebook oder CVS) Rücksicht genommen. Das Risiko und der damit verbundene Zeitaufwand ist aber relativ klein.

Risiko-ID	Beschreibung	Auswirkungen	Massnahmen	Begrenzung des Schadens	Aufwand	Zeitverlust bei Eintritt	p(Eintritt)	Gew. Schaden
R01	Ein Mitarbeiter fällt aus(Krankheit, Unfall, ..)	Eingeschränkte Produktivität	Partner muss jederzeit übernehmen können	-	15	80	0.01	0.8
R02	Klagen wegen illegaler Netzbenutzung	Imageschaden für MA und HSR	Rechtslage abklären	Rechtsschutz beantragen	1	5	0.05	0.25
R03	Ein Notebook fällt aus	Eingeschränkte Produktivität	-	Fremdnotebook organisieren	0	16	0.05	0.8
R04	Connection time während der Busfahrt ist zu klein	Es entsteht keine Verbindung	Frühe Tests	Auf Tram umsteigen, ev. Aufgabenstellung anpassen	4	4	0.20	0.8
R05	Zu wenig offene Netze	Zu wenige Verbindungen	-	Anderes Testgebiet suchen	4	8	0.20	1.6
R06	Ausfall des CVS Servers	Dokumente müssen von Hand zusammengefügt werden	Backup	Alternativen CVS Server aufsetzen	2	2	0.05	0.1

Abbildung 14.1: Risikoanalyse

Kapitel 15

Projektmanagementplan

15.1 Einführung

15.1.1 Projektübersicht

Wardriving ist ein unter Geeks beliebtes Hobby. Dabei wird versucht mit einem Wirelessfähigen Gerät, meist einem Laptop, über ein fremdes Drahtlosnetzwerk eine Verbindung ins Internet aufzunehmen. Dieser Ansatz ist sehr statisch. Gelingt es einem nämlich einmal eine Verbindung aufzubauen, ist man geographisch an einen Ort gebunden, um sicherzustellen, dass man die Verbindung nicht wieder verliert.
Genau hier setzt der Warsurfer an. Mit diesem Programm wird es möglich mehrere offene Drahtlosnetzwerke, die einem beispielsweise während einer Busfahrt durch Zürich begegnen, auszunutzen um quasi eine stetige Verbindung ins Internet zu erhalten.

15.1.2 Projekt Lieferumfang

Nachfolgend sind die Projektanforderungen laut Aufgabenstellung aufgelistet. Diese sind in vollem Umfang zu erfüllen: Als Produkt der Arbeit soll eine funktionsfähige Software vorliegen, die zu demonstrieren ist. Die Software muss direkt auf der entsprechenden Hardware ausführbar sein, sowie bei Verwendung der dokumentierten Entwicklungs-Tools auch generiert werden können. Für die eingesetzte Plattform ist eine vollständige Installationsbeschreibung bereitzustellen. Des weiteren wird eine Bedienungsanleitung erwartet (Form frei wählbar). Ergänzend soll ein nach den unten stehenden Anforderungen aufgebauter Bericht entstehen. Abzugeben sind zwei CD-ROM's, welche die Software und den Bericht in elektronischer Form enthalten (PDF-Dateien und Originaldateien des benutzten Textsystems). Der Bericht ist zusätzlich zweimal in gedruckter Form mit Ringbindung (kein Ordner!) bereitzustellen.
Geforderte Berichtsinhalte:

1. Resultatdokumentation: Sie beschreibt alle Resultate der Arbeit und soll projektbegleitend gemäss dem für die Arbeit gewählten Vorgehensmodell erstellt werden. Sie soll alle wichtigen Informationen enthalten, die ein Ingenieur benötigt, der die Arbeitsresultate ohne vorgängige Kenntnis des Projekts weiterverwenden möchte.

2. Projektdokumentation: Sie umfasst jegliche Dokumentation, die sich auf die Durchführung der Studienarbeit bezieht. Dazu gehören auch ein nachgeführter Projektplan (Arbeitspakete, Zeitplan mit Meilensteinen, Plan- und Ist-Aufwände), Kurzprotokolle aller Besprechungen und ein Projektschlussbericht (was wurde erreicht bzw. nicht erreicht, was ist in der Durchführung gut/schlecht gelaufen, Schlussfolgerungen). Im Bericht ist die Aufteilung der Arbeit innerhalb der Gruppe auszuweisen.

Neben dem oben geforderten Gesamtbericht sollen folgende Berichtsteile zusätzlich in englischer Sprache als separate elektronische Dokumente bereit gestellt werden:

- Technischer Bericht
- Systemdokumentation
- Benutzerdokumentation
- Projektdokumentation (Software-Engineering-Dokumente)

Im weiteren soll die Gliederung und der Inhalt des Berichts den Regelungen der Abt. Informatik entsprechen.
Teildokumente sollen derart in den Bericht integriert werden, dass ein hierarchisches Gesamt-Inhaltsverzeichnis mit durchgängiger Seitennummerierung ermöglicht wird!
Datenblätter, Handbücher und allfällige Kopien anderer Dokumente, die nicht selbst erstellt wurden, aber für weitere Arbeiten mit den Systemen nützlich sind, sollen separat zum Gesamtbericht abgegeben werden (Abgabeform frei)
Berichtsinhalte, die nicht selbst erarbeitet wurden, sind mit ihrer Quelle zu bezeichnen.
Das Projekt wird in 2 Phasen durchgeführt. In der ersten Phase wird ein Prototyp erstellt, mit welchem es möglich sein wird Messungen der Verbindungsdauer und deren Qualität durchzuführen. Diese Messungen sollen ebenfalls Bestandteil des Schlussberichtes sein.

15.1.3 Referenzliste

15.2 Projekt Organisation

15.2.1 Prozess Modell

Offizieller Projektstart: 24.10.2005
Offizielles Projektende: 16.12.2005

Die Grundlage des Projektes bildet der in der ersten Iteration erstellte Prototyp, mit welchem eine eingehende Analyse der Umgebung ermöglicht wird. Der Prototyp wird nach der ersten Iteration, am 08. November 2005 für Messungen zur Verfügung stehen.
In einer nachfolgenden zweiten Iteration wird anhand der Messungen der Routing-Teil des Warsurfers erstellt.

15.2.2 Organisation: Abgrenzung und Schnittstellen

Für die Überwachung der technischen Fortschritte, sowie der Dokumentation ist der betreuende Dozent Prof. Dr. A. Steffen zuständig. Die Qualität des Produktes wird durch regelmässige Meetings, in denen die für den Projektverlauf relevanten Entscheidungen, sowie die nächsten Schritte besprochen werden, sichergestellt und durch Reviewsitzungen am Ende einer Projektphase verstärkt.

15.2.3 Projektverantwortlichkeiten

Prof. Dr. A. Steffen: Verantwortlicher Dozent
Oliv Schacher: Analyse, Design, Implementation, Dokumentation
Cedric Wider: Analyse, Design, Implementation, Dokumentation

15.3 Projektmanagement

15.3.1 Management Ziele und Prioritäten

Für dieses Projekt wird ausschliesslich OpenSource Software eingesetzt.
Geleistete Arbeitsstunden, sowie Tätigkeiten, die während dieser Zeit ausgeführt worden sind, werden textuell erfasst.
Die Projektdauer beträgt 7 Wochen, wobei der Wochenaufwand bei mindestens 40 Stunden liegt.
Zu Beginn des Projektes wird ein Prototyp erstellt, welcher es ermöglicht Messungen der Umgebung durchzuführen. Die Entscheide, die aufgrund dieser Resultate gefällt werden, fliessen direkt in die Konstruktion ein.

15.3.2 Voraussetzungen, Abhängigkeiten und Randbedingungen

Das Projekt soll auf Sourceforge gehostet werden.
Der Code wird in Python geschrieben. Die Studenten eignen sich die Kenntnisse dieser Skriptsprache selbständig an, da dies nicht im Lehrplan enthalten ist.
Das Projekt wird im Rahmen einer Diplomarbeit durchgeführt.

15.3.3 Risiko Management

Für dieses Projekt wird eine detaillierte Risikoanalyse erstellt, welche die Verfolgung der auftretenden Risiken, sowie die zu treffenden Massnahmen bei Auftritt des entsprechenden Risikos enthält.

15.3.4 Überwachung und Kontrolle

Nach jeder Phase des Projektes wird eine Review-Sitzung stattfinden, bei welcher die ausgeführten Arbeiten bewertet und ein Ausblick und Nötige Schritte

besprochen werden. Alle Arbeiten werden dokumentiert.

15.4 Entwicklungsprozess

15.4.1 Methoden, Tools und Techniken

- Da RUP sich mit seinen Phasen und enormem Dokumentationsaufwand für dieses Projekt nur bedingt eignet, wird nicht strikt nach RUP vorgegangen. Wie bereits erwähnt, wird als erstes eine Analyse durchgeführt, deren Ergebnisse starken Einfluss auf den Design haben wird.

- Editoren: kate, kile, gedit, vim und andere standard-Linux Editoren
- Entwicklungsplattform: Debian GNU/Linux
- Programmiersprache: Python, plsql, SQL
- Alle Worksproducts (Code, Dokumentation) werden in einem CVS Repository verwaltet
- Coding Standards: Jede Klasse und jede Methode wird kurz dokumentiert. Es ist ausserdem auf übersichtlihen Code zu achten.

15.4.2 Software Dokumentation

Neben der Codedokumentation sind folgende Dokumente zu erstellen:

- Genaue Installationsanleitung
- Benutzerdokumentation in deutscher und englischer Sprache

15.5 Entwicklungsplan

Es ist geplant die Entwicklung des Warsurfers in zwei Iterationen zu erstellen. Dabei soll in der ersten Iteration die Netzsuche, sowie der Verbindungsaufbau mit den verfügbaren Netzen realisiert werden. Die zweite Iteration befasst sich mit dem Roaming über verschiedene Netze und soll es ermöglichen Downloads auszuführen ohne bei einem Netzwechsel die Verbindung zu verlieren.

15.5.1 1. Iteration

Die erste Iteration befasst sich, wie bereits erwähnt, mit dem Verbindungsaufbau und der Netzsuche. Um in Erfahrung zu bringen, mit welchen Kenngrössen zu rechnen ist, wird zu Beginn eine Analyse durchgeführt, deren Erkenntnisse dann in den Design einfliessen werden. Schliesslich wird der Design in ausführbaren Code übergeführt, welcher zum Schluss der Iteration getestet wird.

Analyse

Die Analyse soll eine Grundlage liefern, welche es ermöglicht einen Design nach vorherrschenden Kriterien auszuführen. Dabei soll untersucht werden welche Wireless Sniffer sich am besten für den Einsatz eignen. Ausserdem soll ermittelt werden mit welchen Verbindungszeiten, respektive Verfügbarkeitszeit während einer Bus-, Zug-, oder Tramfahrt zu rechnen ist. Im Design ist ein Algorithmus zu entwickeln, welcher sich mit der Suche, bzw. der Auswahl des richtigen Netzes beschäftigt. Für diesen Algorithmus sind die Eckwerte zu analysieren. So sollen Situationen aufgezeigt werden, die auftreten könnten.

Design

Der Design wird nicht strikt nach RUP durchgeführt. Es werden jedoch die nötigen Designdokumente wie Klassendiagramm oder Domainmodell erstellt. Ausserdem wird ein Designdokument erstellt, welches den Design erklärt und die getroffenen Designentscheide begründet.

Coding

In dieser Phase wird der ausführbare Code erstellt. Schwerpunkt dabei ist, dass der Code leserlich bleibt und ausreichend dokumentiert ist. Es soll einer Drittperson möglich sein, den Code zu verstehen und diesen zu erweitern. Die Codedokumentation wird mit dem Tool PyDoc erstellt und dem Schlussbericht beigefügt.

Testing

Um die erstellte Software zu testen muss eine Testumgebung eingerichtet werden. Es muss mit dieser Umgebung möglich sein, eine Fahrt entlang diversen Accesspoints zu simulieren um feststellen ob es möglich ist eine Verbindung mit diesen Accesspoints herzustellen. Dabei müssen folgende Punkte beachtet werden:

- ESSID
  Die ESSID muss einem bestimmten Namensschema entsprechen. Dadurch wird sichergestellt, dass Statistiken, welche sich auf die in der realen Welt aufgezeichneten Daten beziehen nicht durch simulierte Netze verfälscht werden.

- Strength
  Die Stärke soll variieren. In der realen Welt wird dies von einem bewegten Betrachter aus ebenfall so sein. Wird der Accesspoint zum ersten mal entdeckt, wird sich dieser in etwas grösserer Entfernung befinden. Während der Fahrt kommt dieser Accesspoint näher und somit wird auch die Signalstärke zunehmen und beim Entfernen vom Accesspoint wieder abschwächen.

- Zeit
  Je nach dem wie schnell man sich an einem Accesspoint vorbei bewegt oder wie weit dieser entfernt ist, ist dieser länger oder weniger lang für den Betrachter sichtbar. Simulierte Netze sollen in zeitlich variabler Länge

sichtbar sein. Ausserdem bietet eine zeitliche Varianz beim Entwickeln den Vorteil, dass das Debugging vereinfacht werden kann, indem Netze über längere Zeit sichtbar gehalten werden und somit der Wechsel auf ein neues Netz verzögert wird. Dies ermöglicht es einem Entwickler die automatisch gemachten Einstellungen von Hand zu verifizieren.

15.5.2 2. Iteration

In der zweiten Iteration wird sichergestellt, dass eine bestehende Verbindung ins Internet, während einem Wechsel von einem Accesspoint zu einem anderen, nicht abbricht. Es soll also eine mehr oder weniger stetige Verbindung ins Internet simuliert werden.

Analyse

In der Analyse sollen Möglichkeiten aufgezeigt werden, welche die Erreichung des oben beschriebenen Ziels möglich machen. Dabei muss darauf geachtet werden, dass der Zeitaufwand für die Implementation dieser Möglichkeiten im Rahmen der Diplomarbeit bleibt.
Die Aufgabenstellung besagt, dass es möglich sein muss, Emails während eines Rides zu lesen. Es muss also ein Mechanismus gefunden werden, wie dies zu bewerkstelligen sei.

Design

Der Design verläuft analog zu demjenigen der Iteration 1.

Coding

Das Coding verläuft analog zu demjenigen der Iteration 1.

Testing

Um die getroffenen Massnahmen zu testen ist ebenfalls wieder eine Testumgebung nötig. Idealerweise kann mit der aus Iteration 1 bestehenden Umgebung gearbeitet werden. Dabei wird ein Ride simuliert und Verbindungen aufgebaut, welche über die verschiedenen Netze mit unterschiedlichen IP-Einstellungen erhalten bleiben muss.

15.5.3 Abhängigkeiten

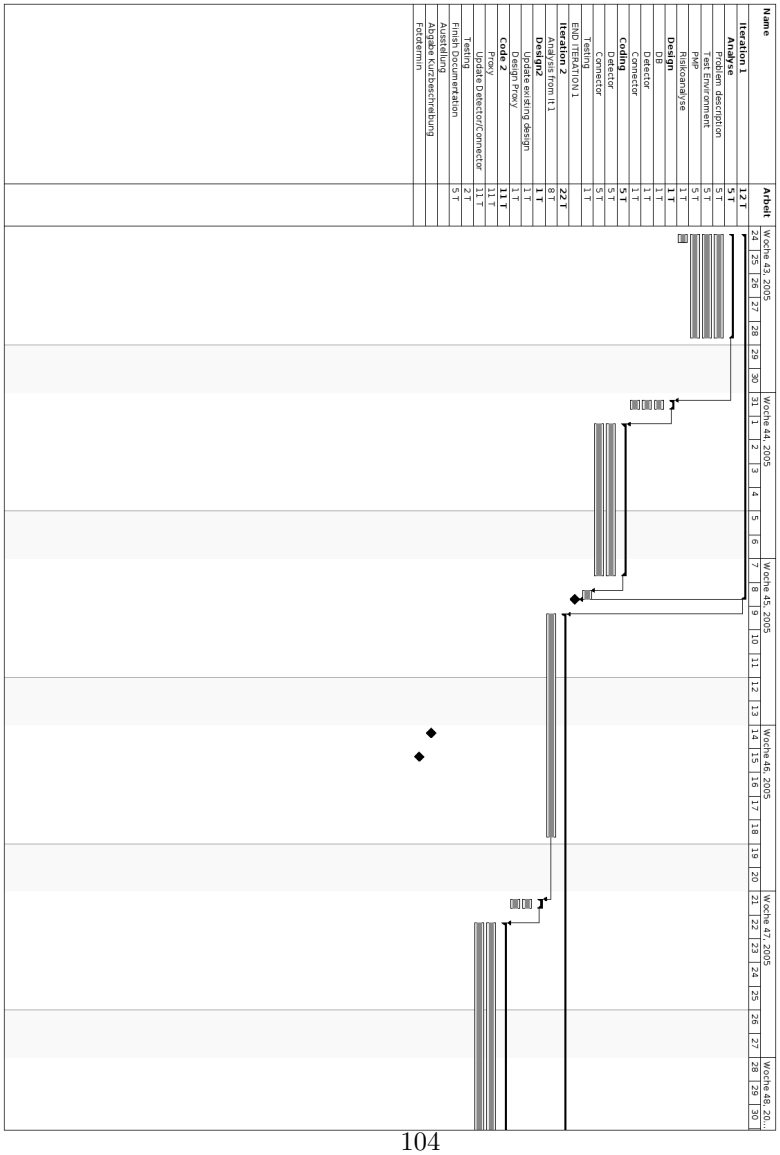
Der Roamer, also die Einheit im Warsurfer, welche sich um die Aufrechterhaltung der Internetverbindung kümmert, hängt stark vom darunterliegenden Connection Management ab, weshalb dieser auch in einer zweiten Iteration implementiert wird.

15.5.4 Termine

Alle Termine sind im Projektplan ersichtlich.

Kapitel 16

Projektplan



Woche 49, 2005
Woche 50, 2005
Woche 51, 2005
Woche 52, 2005
Woche 1, 2006
Woche 2, 2006

Literaturverzeichnis

[1] The Shmoo Group. Airsnort. http://airsnort.shmoo.com/, 2005.

[2] Gerald Combs u.a. Ethereal introduction. http://www.ethereal.com/introduction.html, 2005.

[3] Ahmad Baitalmal u.a. What is wifi radar? http://www.bitbuilder.com/wifi_radar/, 2004.

[4] Christian Weiske. Wlanscanner. http://www.cweiske.de/phpgtk_apps.htm#wlanscanner, 2005.

[5] "Dragorn". Kismet readme. http://www.kismetwireless.net/documentation.shtml, 2005.

[6] Radionet Ltd. Wlan radio system. http://www.radionet.com/_FileRoot/318029.pdf, 2003.

[7] Jim Geier. 802.11 mac layer defined. http://www.wi-fiplanet.com/tutorials/article.php/1216351, 2002.

[8] Inc. Javvin Technologies. Wlan: Wireless lan by ieee 802.11, 802.11a, 802.11b, 802.11g, 802.11n. http://www.javvin.com/protocolWLAN.html, 2005.

[9] Prof. Ursula Sury. Rechtliche aspekte von offenen accesspoints. http://www.sgrp.ch/pages/download/Recht_Wireless.pdf, 2005.

[10] James D. Solomon. *MOBILE IP - The Internet Unplugged*. PRENTICE HALL, 1998.

[11] Oliver Ertl u.a. Openwrtdocs. http://wiki.openwrt.org/OpenWrtDocs, 2005.

[12] et al. R. Fielding. Hypertext transfer protocol – http/1.1, rfc 2616. http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35, 1999.

[13] OpenVPN Solutions LLC. Openvpn 2.0 howto. http://openvpn.net/howto.html, 2005.

[14] Tony Gentilcore. Fasterfox - performance and network tweaks for firefox. http://fasterfox.mozdev.org/, 2005.

[15] Jouni Malinen. Host ap driver for intersil prism2/2.5/3, hostapd, and wpa supplicant. http://hostap.epitest.fi/, 2005.

Glossar

A

Address Resolution Protocol (ARP) Netzwerkprotokoll, das die Zuordnung von Internetadressen zu Hardwareadressen möglich macht. Obwohl es nicht auf Ethernet- und IP-Protokolle beschränkt ist, wird es fast ausschließlich im Zusammenhang mit IP-Adressierung auf Ethernet-Netzen verwendet. ARP gehört zur Internetschicht der TCP/IP-Protokollfamilie.[Wikipedia].

Application Programming Interface (API) Schnittstelle, die von einem Betriebssystem oder von einem anderen Softwaresystem weiteren Programmen zur Verfügung gestellt wird.[Wikipedia].

B

Basic Service Set Identifier (BSSID) TODO... Macadresse eines APs.

C

Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) Verfahren für den Zugriff mehrerer Netzwerkstationen auf dasselbe Übertragungsmedium. Es wird häufig u.a. bei drahtlosen Netzwerken (Wireless LANs) eingesetzt, findet aber abgewandelt auch bei Technologien wie ISDN Anwendung.[Wikipedia].

Channel Hopping Schnelles Wechseln zwischen den verschiedenen Wireless-Kanälen.

D

Debian GNU Linux (Debian) Debian ist eine frei erhältliche Linux Distribution, welche asuschliesslich unter der GPL (GNU Public License) Entwickelt worden ist. Debian beeindruckt vor allem durch den Einsatz des Paketmanagement-Tools ´´apt-get".

Domain Name Server (DNS) Dienst im Internet, welcher Hostnamen in IP-Adressen auflöst.

Dumbnet (dnet) Netzwerkbibliothek, welche die manipulation von IP-Paketen, Adressen, Routingtables etc. erlaubt.

E

Extended Service Set Identifier (ESSID) Name für ein Wireless netzwerk.... TODO, define!

F

Firmware Software, welche sich in einem ROM (Read only Memory) befindet.

G

Geek Eine Person, die sich gerne mit Technologie, Computern und neuen Medien beschäftigt. Diese Definition von Geek ist nahe verwandt mit der klassischen Definition von Hacker. [Wikipedia].

Graphical User Interface (GUI) Eine Grafische Benutzeroberfläche ist die grafische Schnittstelle auf Computern, die eine Interaktion mit dem Benutzer verlangen.

H

HTTP Pipelining (Pipelining) Mehrere HTTP Requests über die selbe TCP Connection abwickeln.

I

ifconfig (ifconfig) Kommando um Netzwerkinterfaces zu konfigurieren.

iwconfig (iwconfig) Kommando um Wirelessinterfaces zu konfigurieren.

L

Link Prefetching (Prefetching) Herunterladen von Webseiten, die der Benutzer vermutlich in naher Zukunft anschauen wird.

M

MAC Adresse (MAC) Die MAC-Adresse (Media Access Control, auch LAN-Adresse genannt) ist die Hardware-Adresse vieler Netzwerkgeräte (vor allem Netzwerkkarten), die zur eindeutigen Identifikation des Geräts im Netzwerk dient.

Message Digest 5 (MD5) Hashfunktion, welche einen Text beliebiger Länge in einen 128-bit Hash umwandelt.

N

Network Address Translation (NAT) Routerfunktion um mehrerer Rechner an ein IP-Netzwerk mit nur einer offiziellen IP-Adresse anzubinden.

P

Plugin (Plugin) Software, welche eine andere Software (z.b. Browser) um eine weitere Funktion erweitert, z.b. das Anzeigen neuer Medientypen.

Proxy Allg. Zwischenspeicher für Daten. Häufig verwendet werden HTTP-Proxies, welche Webseiten und Bilder speichern um diese bei einem nächsten Aufruf schneller zu liefern. Es gibt aber auch Proxies für andere Protokolle (DNS, POP, ...).

PyDoc (PyDoc) Standard Dokumentationstool für Python Code.

R

Ride (Ride) Eine einzelne Fahrt durch ein bestimmtes Gebiet während der Warsurfer eingeschaltet ist und Daten über Netze sammelt.

S

scapy (scapy) Paketgenerator und Decoder für Python.

Secure Shell (SSH) Secure shell oder SSH ist sowohl ein Programm als auch ein Netzwerkprotokoll, mit dessen Hilfe man sich z.B. über das Internet auf einem entfernten Computer einloggen und dort Programme ausführen kann. Die IANA hat dem Protokoll den Port TCP:22 zugeordnet.[Wikipedia].

Service Set Identifier (ssid) Als Service Set Identifier (SSID) oder auch Network Name bezeichnet man die Kennung eines Funknetzwerkes, das auf IEEE 802.11 basiert. [Wikipedia].

W

Wired Eqievalent Privacy (WEP) Wired Equivalent Privacy (WEP) ist der ehemalige Standard-Verschlüsselungsalgorithmus für WLAN. Er soll sowohl den Zugang zum Netz regeln, als auch die Integrität der Daten sicherstellen. Bereits 2001 wurde der Schlüssel geknackt, weshalb WEP heute auch als ´´Wireless Equivalent Placebo" bezeichnet wird.

Wireless Sniffer (Sniffer) Ein wireless Sniffer ist ein Programm welches in der Lage ist, eine Wireless-Karte in einem System auf diese Art zu konfigurieren, dass es möglich ist, Pakete, die über einen AccessPoint verschickt werden, mitzuhören.

Abbildungsverzeichnis

2.1 Warsurfer Theorie [Schematisch] 6

4.1 802.11 Standards 10
4.2 802.11 Frequenzen 11
4.3 Wireless Basics 12

6.1 Vergleich der Wireless Sniffer 18
6.2 Die verschiedenen Assoziationsstati 21
6.3 802.11 Frames detailliert 21
6.4 Funktionsweise des Home Proxy 28
6.5 MobileIP 29
6.6 Verwendete Hardware und Treiber 30

7.1 Interprozesskommunikation über die Datenbank 34
7.2 Klassendiagramm 35
7.3 Verbidungsalgorithmus 39
7.4 Verbindung mit dem Internet 40
7.5 Mailtransfer über HTTP 44
7.6 Roaming mit OpenVPN 45

8.1 Installer: Einführung 59
8.2 Installer: Datenbankname 59
8.3 Installer: Datenbankbenutzer 60
8.4 Installer: Datenbankpasswort 60
8.5 Installer: Detectorinterface 61
8.6 Installer: Connectorinterface 61
8.7 Kismet Screenshot 69
8.8 Detector Screenshot 69
8.9 Connector Screenshot 70
8.10 Download Daemon Screenshot 71
8.11 Mailfetcher Screenshot 71
8.12 Download Client Screenshot 72

12.1 Der zukünftige Warsurfer? 91

14.1 Risikoanalyse 96